DTIC

CREATING A MOBILE AUTONOMOUS
ROBOT RESEARCH SYSTEM (MARRS)

THESIS

Thomas E. Clifford, First Lieutenant, USAF
Hubert G. Schneider, Captain, USAF

AFIT/GE/ENG/84D-19

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/84D-19

CREATING A MOBILE AUTONOMOUS ROBOT RESEARCH SYSTEM (MARRS)

THESIS

Thomas E. Clifford Hubert G. Schneider III
First Lieutenant, USAF Captain, USAF

AFIT/GE/ENG/84D-19

AFIT/GE/ENG/84D-19

CREATING A MOBILE AUTONOMOUS ROBOT RESEARCH SYSTEM (MARRS)

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Accession For		
NTIS GRA&I		X
DTIC TAB		☐
Unannounced		☐
Justification		
By		
Distribution/		
Availability Codes		
Dist	Avail and/or Special	
A-1		

Thomas E. Clifford

First Lieutenant, USAF

Hubert G. Schneider III

Captain, USAF

December 1984

Approved for public release; distribution unlimited

Preface

The creation of MARRS-1 and the AFIT Mobile Robotics Laboratory was made possible through the energetic support of many individuals and organizations. In particluar, we want to thank our thesis advisor Dr. Matthew Kabrisky and the Department of Electrical Engineering for providing the ultimate research environment where opportunities for self realization and true contributions to science and technology flourish. Additionaly we thank Captain Rob Milne, Captain Lee Baker, Robert Durham, Orville Wright, Dan Zanbom, Dick Wager, Charlie Powers, Stan Bashore, and Mary Elizabeth Dennis of AFIT/ENG; Leroy Cannon, and Steve Coats of AFIT/ENY; Ron Gabriel of AFIT/ENP; Carl Shortt, Ron Ruley, and Russell Murry of AFIT/RMF; our sponsor, Tim Anderson of AFMRL; Lt. Keith Carter of AFWAL/AADM-2; Amn. John Burns of AFIT/PA; SSgt. Robert Lofland of Det. 2 1361 AV Sq; Doug Sauer of SRL; Tom Meyer and Gordon Doughman of Motorola; Dick Parr, Abe Siff, and Clair King of Datametrics; and Dennis Van Deusen of Virtual Devices.

Many thanks also go to Lt. Randall Owen who started this new endeavor at AFIT and Captain James Holten, an AFIT Phd. candidate, for his suggestions and encouragement.

But most of all, we are especially grateful to our parents, wives, and children whose support and sacrifices have allowed this project to be "more than just a thesis."

Contents

List of Figures

Figure		Page

List of Tables

Abstract

The Mobile Autonomous Robot Research System (MARRS-1) was created as the first of a series of autonomous vehicle prototypes for the Air Force Institute of Technology. The major accomplishment in developing MARRS-1 is the integration of Optical Shaft Encoder (OSE) data with Ultrasonic Sonar range and direction information to produce accurate environmental maps that are relative to the robot. The OSE and Sonar subsystems make up the most important part of MARRS-1's Navigation Computer. With these two subsystems and minimal additional software, mapping and obstacle avoidance become a reality. The thesis includes schematics, parts list, and software listings for the MARRS-1 Navigation Computer. Additionally, the mapping and navigation algorithmsare shown implemented in the BASIC language with numerous example graphics maps created by the integrated MARRS-1 robot. Issues involved in solving mobile robotics problems are discussed. Originator-supplied key words included:

p.1473(last p.)

CREATING A MOBILE AUTONOMOUS ROBOT RESEARCH SYSTEM (MARRS)

I. Introduction

Background

A Nuclear, Biological, Chemical (NBC) contaminated environment presents a severe hazard for aircraft ground maintenance crews. A mobile autonomous robot capable of performing simple aircraft maintenance tasks could protect many lives from NBC exposure while allowing the Air Force to maintain its combat readiness.

In order for a mobile autonomous robot to function properly on a flight line, the robot must perform the following tasks:

1) Avoid obstacles (both static and dynamic),
2) Locate the appropriate aircraft,
3) Perform fueling and/or maintenance,
4) Return to previous work station.

Early AFIT efforts in mobile robotics were hampered by a lack of suitable hardware and software support (1:59-66). The acquisition of a Heathkit Hero-1 robot, laser barcode reader, and additional sonar sensors by 1Lt Randall J. Owen II for his thesis (2:I-3) was a great move forward for AFIT in mobile robotics. Before any autonomous vehicle can fuel its first aircraft there are however, several problems which

must be overcome. These problems may seem trivial but in fact are the limiting factors in mobile robotics today. Many of these problems are found in the Hero-1 as well as other robot systems:

1) Lack of robot software,
2) Combined error effects in the robot,
3) Reduced robot efficiency due to the dedication of resources for error compensation,
4) Understanding and reducing the amount of performance data.

Objective

This thesis is a follow-on effort to Lt Owen's thesis: "Environmental Mapping by a Hero-1 Robot Using Sonar and a Laser Barcode Scanner". The main objective of this thesis is to continue the development towards an autonomous robot capable of moving from one location to another while avoiding obstacles. However, the thesis effort is divided into the following sub-objectives:

1) Identify and eliminate design deficiencies in the AFIT HERO robot.
1) Establish a software development capability for the AFIT HERO robot.
2) Establish a library of robot programming software and algorithms.

3) Create and conduct tests that will identify system errors and provide figures of merit on robot performance.

4) Establish an AFIT mobile robotics laboratory.

Why MARRS-1

The HERO-1 robot has many design deficiencies (chapter III) that make it difficult to continue its use as a research robot beyond what has been done by Lt. Owen. The two greatest deficiencies of the HERO-1 are the front wheel design and the lack of online and offline software development capabilities. The ability to rapidly develop and debug software was seen as one of the most important improvements needed in the AFIT mobile robotics research environment. HERO-1 robot programmers are required to write programs in machine language, assemble the code by hand, then enter the code as hexadecimal numbers from a keypad on the robot. Programs once loaded on the robot may then be saved, for later use, on audio cassette at 300 baud. A better approach to mobile robot programming is necessary.

Although the HERO-1 is well endowed with sensors (a single three degree beamwidth ultrasonic sonar range finder, a 256 level light intensity sensor, a 256 level sound volume sensor, an ultrasonic motion detector, and wheel odometer) ,its abundant selection of sensors is thwarted by its lack of useful and user friendly software.

To make things worse, the HERO-1 robot operates in an open loop (no active feedback) mode. A quality mobile research robot was needed but could not be found commercially. Two options were available:

1) Build a mobile robot from scratch.

2) Improve the existing HERO-1 design.

The second option was taken resulting in a robot named Mobile Autonomous Robot Research System (MARRS)-1. This also resulted in the creation of the AFIT Mobile Robotics laboratory which initially came more as a necessity rather than as a conscious effort.

AFIT's mobile robotics research was being conducted in the Electrical Engineering Department's Signal Processing laboratory. Work space there was at a premium. A search for lab space and a mobile robotics test range resulted in moving the research effort to a building needing rehabilitation. Over a period of months of scrounging for equipment, furniture, and supplies, the new mobile robotics work area was in fact a laboratory. AFIT building renovation plans have identified 800 square feet for mobile robotics research.

II. Background Issues of Mobile Robotics

Basic Requirements for a Mobile Robot

If a robot is going to move about from one location (present position) to another (the goal), then several important criteria must be met. The robot must know its initial location and heading with respect to some reference point. The location of the goal must also be specified with respect to the reference point or to the robot's own position. Knowledge of the location of all objects around the robot and what the objects are may not be important. However, the locations of obstacles are significant for path planning. An object becomes an obstacle when it becomes apparent that the object lies or will lie along the robot's path towards the goal. This situation is analogous to a human that suffers from agnosia. The person is unable to identify what some objects are but is aware of the object's presence and can maneuver around them. Navigation can be accomplished in a timely manner only if position information can be gathered, coupled together, interpreted, and acted upon.

Four basic elements are required to create a robot that can maneuver its way around objects.

1) A computer (hardware) is required to process sensor data and to provide control logic to the robot subsystems.

2) A drive system is needed for mobility. The most common mobile robot drive systems are composed of digital

controlled DC motors turning wheels since articulate legs are very rare because of their complexity and expense (3:2). Feedback must be provided from the drive system to the computer to give vital information such as acceleration, velocity, distance traveled, and heading.

3) A variety of types of sensors are needed to provide the robot with far and near range obstacle detection, object identification, and sensory system cross-checks. These sensors, coupled with the drive system feedback give the computer the information necessary to maintain an accurate estimation of its current position and heading.

4) Computer programs (software) are needed to interpret sensory information, plan/execute required operations, and control robot subsystems.

Computer Hardware

Until a few years ago autonomous vehicles were unheard of in real life. The advent of the microcomputer has made fact possible out of fiction.

The two most important uses of a microcomputer in robotics are device control and data processing. In a dynamic environment, multiple computers may be employed to process data and control devices.

Several prototype robots with multiple computers operate under a master-slave principle. Generally the motion controllers act as slaves to the sensory

processors/interpreters. Communication between masters and slaves is handled by an interrupt scheme (4:212).

Computational Element Selection

Computational elements in a robot can be anything from a hydraulic valve to a multiuser/multiprocessor mainframe computer. Fornow, the discussion centers on digital microprocessors since most present day robotics projects are so based. It should be be remembered, however, that there are many alternatives available for fulfilling a robot's computational needs.

Microprocessor Features of Interest

Although the following list of microprocessor features is not complete nor exhaustive in identifying a robot designer's needs, it does point to those features needing greatest consideration when choosing a microprocessor for an imbedded robotics computer application. The features are:

1 Existing software support;

2 Interrupt structure:

- a. Nonmaskable interrupt;
- b. Maskable interrupt;
- c. Software or System interrupt;
- d. Priority interrupt;
- e. Buffered interrupts when masked;

3 Input/Output facilities;

4 Instruction set and addressing modes;

5 A Test and Set instruction for resource control;

6 Addressable memory space;

7 Hardware support chips:

   a. DMA controller;

   b. Memory management;

   c. Numeric coprocessor for floating point operations;

   d. I/O;

8 Special supervisor modes for system failsafe monitoring;

9 Processor compatibility with supportchips/devices;

10 Minimum system and support cost; and

11 Development team familiarity with the chosen microprocessor.

Existing software to support the chosen microprocessor has to be the single most important factor used in the selection process since 90% of project costs historically have been for software development. By procuring commercial support software, in-house software development overhead is decreased and more resources can be directed toward robot specific software. Development team familiarity with a given microprocessor has been an overriding choice factor in a number of projects such as in the development of the Motorola 6808 based HERO-1 robot.

In reviewing the presently available microprocessors against the above criteria, the Z80 microprocessor is the best choice (here at AFIT) mainly because of available hardware/software support. However, when not considering available support, the MC68000 family of microprocessors has the best feature set for robotics applications because (5,6,7):

1 Its memory oriented architecture (as opposed to register oriented) facilitates memory to memory transfers of data which is very useful for sensory data processing.

2 Sixteen Megabytes of memory may be directly addressed allowing for a mobile robot to carry with it a tremendous amount of instructions and data without the need for an external mass storage device.

3 A test and set instruction is provided for controlling access to shared resources in a multiprocess, multiprocessor, and/or interrupt driven environment typical of robot designs and applications.

4 Synchronous or asynchronous interfacing to external devices allows great flexibility to the hardware designer.

5 Any combination of processor registers may be saved or restored with a single instruction.

6 All program code is directly relocatable since all address references are added to a base register that is set to effect the relocation.

7 The UNIX operating system is widely used for MC68000 systems and has with it some very powerful features. However, the overhead in terms of memory usage and speed of execution may make this a negative feature for mobile robotics. Also, although UNIX is multiuser/multitasking, it is not optimized for the realtime computational environment required by robotics. It is, however, a very powerful operating system for software development.

Today's choice in the use of a particular microprocessor for a robotics application may not be tomorrow's choice. Designers must plan for an upward migration of requirements and capabilities and yet accomplish tomorrow's job today.

Mechanical Drive

A typical drive system of a robot utilizes a computer controlled DC motor capable of variable speeds in both forward and reverse directions. Motors are mounted on one or more wheels to provide steering and motion control.

Mechanical Device Performance (8:12-24)

Software routines that control mechanical devices must be designed with the following items in mind:

1 Servo systems must be stable over wide dynamic ranges.

2 Spatial resolution has finite errors (mechanical accuracy) and errors due to quantization.

3 Repeatability is more difficult when the mechanical accuracy of the robot is less than the accuracy of the sensory data used to modify the robot's action.

4 Repeatability will suffer from component wear and aging.

5 Errors will accumulate. Sensory data must be used to update mechanical device commands.

6 Compliance to a command may be dependent upon the mechanical loading of system components.

7 Compliance to a command may exhibit a hysteresis effect depending on the direction of arrival (such as an accumulated 360 degree turn from the right is really 365 degrees but coming from the left it is only 355 degrees).

8 Oscillations may occur if command compliance is too slow.

Sensors

Without sensors, a robot is nothing more than a Numerically Controlled (NC) machine (8:57). Sensors allow a robot's actions to be determined based upon the robot's sensory interpretation of the world.

Before a robot can move about and perform any function, it has to sense its surroundings. This information will allow it to compute a clear path of travel and prevent it from bumping into obstacles.

There are numerous types of sensors commercially available today for object detection and identification. These sensors range from the simplest contact switch to the most complex vision systems. Yet each sensor has its own limitations.

For example, a microswitch can be used to detect whether or not an object has made contact with the robot. However, it can not tell how much pressure is on the object nor the maximum amount of pressure that can be tolerated. A fragile item such as a flower might be crushed by a robot hand that lacks very low pressure touch sensors.

The sensing of position can be simulated through optical devices (9:71). The use of photo emitters and receptors can measure intensity of light as well as touch. Another method of simulating touch is through membrane contact surfaces. Many of these membrane sheets form an array of switches. The greater the number of switches used, the better the resolution will be. However, increasing the number of sensors means an increased amount of data which requires much more computing time (10:73). This holds true for sensors in any domain and not just touch sensors.

The sensors discussed thus far will only give the robot information about its immediate surroundings. If the robot is to maneuver around in some unknown environment then it must also have access to sensory information about objects at much greater distances. What is required is some form of vision.

Robotic Vision

Moravec sized up the state of the art in robotic vision by saying, "There's a handful of techniques for robot vision that sort of work, but none that works spectacularly well. We're all still groping in the dark (11:73)." As robotic vision goes, so goes mobile robotics.

Jarvis (12) points out that the human visual system uses many techniques to extract range information about the objects in the field of view and which may be used by robots. These include; changes in brightness level, binocular convergence (the inward pointing of the eyeballs which is inversely proportional to the range of the object), stereo disparity (the closer the object the greater the disparity), vertical position in the visual field (closer objects are usually lower in the field of view), diminution of size with distance, occlusion clues and outline continuity (complete objects look closer than partially obscured ones).

Range and Direction Techniques

The principal range with direction vision techniques used by robots are triangulation, surface orientation from image brightness, stereo disparity and binocular convergence, ultrasonic sonar, radar, and laser time-of-flight (12).

Triangulation

Triangulation techniques use a narrow beam of light and a camera that is physically displaced from the light beam source. The camera tracks the light beam as it is swept across the field of view. The azimuth and elevation (pointing direction) of the camera and light beam are recorded during the scan. Using Euclidean geometry, a three dimensional picture (3-D) is generated from the camera's 2-D picture and the direction information. Since the camera is physically displaced from the light beam, it can see points in the scene that the light beam can not and vice versa. This lack of common origin causes problems for scene interpretation and is only used in controlled situations.

Controlled Lighting

Surface orientation from scene brightness gives range information indirectly. Using high contrast controlled lighting, edges of objects are defined by looking for a change in brightness between adjacent pixels (picture elements). Keller (13:123) indicates that the generally

accepted minimum pixel count for robotic vision purposes is 256 horizontal by 256 vertical with 256 levels of brightness (grey scale). This pixel brightness comparison creates a tremendous computational burden on a robot and is normally performed remotely by special computers. Once the edges of an object are defined, the relative range of the object is estimated using occlusion clues and outline continuity rules.

Stereo Vision

The stereo disparity and binocular convergence combine the techniques of triangulation and edge detection in that the pixels of two physically separated cameras are matched. The angular position in the field of view of each camera for a specific common point will be different from one camera to the other because of the physical separation of the cameras. The difference between the angular position of the two views is equivalent to the amount of inward turning of the eyes in the human visual system which is inversely proportional to the range of the common point. The greatest problem with this technique is being able to distinguish which pixel of one camera is the same point in the other camera (13:123). This problem can be compounded when the picture being processed has a periodic pattern such as a brick wall or a fence with a vertical or horizontal pattern. In such situations, the computer that is correlating points between pictures gets confused and usually can't solve the problem.

Stereo Convergence in an IC

Iverson (14) describes an integrated circuit (IC) that has 23 pairs of light detectors lined up in two rows with each having its own microlens bonded directly to the chip carrier. The IC is placed behind the lens of a single camera and is capable of doing edge detection and range calculations using stereo disparity and binocular convergence techniques passively without using special lighting. Such an equipped camera system does not suffer from most of the problems of the previous paragraphs and holds great promise for freeing mobile robots from the lab. The accuracy of the system depends on the distance to the object being sensed and the focal length of the imaging lens used on the camera.

Range from Time-of-Flight

Ultrasonic sonar, radar, and laser time-of-flight techniques rely on the propagation of energy (sound, electromagnetic, and light) through air to an object that reflects the energy back. The time to make the trip is proportional to the range of the reflecting object. The problem with these techniques is that the power of the return energy received by the sensor decreases at a rate proportional to the range to the fourth power. Hence, an increase in range coverage requires a tremendous increase in transmitted power which may be beyond safe levels.

Ultrasonic Sonar

Laser light and television systems can provide distance information and much more. However, these systems tend to be very complex and expensive (10:180). A low cost alternative approach is to use an ultrasonic device (like the ones found on the Polaroid Land cameras.)

A high frequency "chirp" is transmitted from the sonar device. A counter keeps track of the time between the instant the signal is transmitted and the time the signal returns. The Polaroid sensor for example has a range of 0.9 feet to 35.0 feet (15:15). Although this sensor can provide the robot with limited information about its surroundings, the object's inclination and geometry could affect the quality of the returned signal. Better results can be obtained with multiple ultrasonic sensors mounted on the robot in different directions (10:183).

Owen (2) showed that mobile robot obstacle avoidance was possible by reducing the amount of information to be processed by the use of multiple ultrasonic sonar sensors giving the range to the closest obstacle at known directions. He also showed that a crude map of the robot's environment could be made from robot position and sonar information.

Jarvis (12:135) indicates that obstacle detection and robot navigation are good applications for ultrasonic sonar because of their low cost and ease of use.

Radar range finders have given way to laser time-of-flight range finders since the beam of the laser is very narrow and can be precisely controlled. Since the path traveled to the object is done at the speed of light, the supporting instrumentation must be capable of 30 nanosecond resolution for range accuracy of 1 centimeter. This requirement pushes the present limits of economical electronics. Jarvis (16) feels that such laser range finders used with standard single camera 2-D vision systems could yield sufficient quality for robotic scene analysis.

Robot Software

Robotics research, development, and applications require advanced engineering and technical skills (8:55). The key element here is computer control via software. The sophistication of the robot's software will vary directly with the complexity of the task to be performed. What distinguishes robot software from other software is the robot's interaction with the real world. The software must account for numerous possibilities and outcomes of situations.

Flexibility thru Resources

Computer software enables a robot to perform a myriad of tasks. However, there is no standard or universal robot programming language. For every unique robot, there is a

unique set of instructions to control that robot.

Flexibility in robot programming is found in the basic operations the robot can perform as determined by its resources. A robot's resources are:

1 algorithms;

2 data;

3 computational power;

4 storage capacity;

5 commandable devices (motors, relays, servos, wheels, arms, end effectors, and other special purpose devices);

6 sensors (gathers or verifies data on system operation or the environment);

7 and master/slave connections with other robots, machines, systems, computers, and/or humans.

Levels of Robot Programming

There are three distinct levels of robot programming: system, task, and operator (8:55). System programming provides the lowest level of routines to control robot resources and interfaces between resources. Task level programming is done as high level calls of the lower system level routines. Operator programming may consist of loading programs into the robot, adding required or optional data, turning the robot on or off, teaching the robot a sequence of operations to be remembered for later use by means of a

"teach box" or "pendant", or physically leading the robot by the hand thru the required operation while having the robot memorize the sequence for later use. Each level of robot programming requires a different level of skill and understanding.

Basic Functions Required

The set of required basic functions for a robot are (8:57):

1 Computation:

 a. analytic geometry is most useful;

 b. coordinate representation and transformation;

 c. vector operations (dot product, cross product, scaling, normalization, and linear operations);

2 Decision (conditional branch based on processed sensor data):

 a. sign test (+, -, Ø);

 b. relation (<, >, =);

 c. Boolean (on, off, true, false, 1, Ø);

 d. bit pattern given a reference pattern;

 e. set operations (member, nonmember, subset, empty set);

3 Communication (internal and external);

4 Movement;

5 Sensor data gathering; and

6 Sensor data processing.

Quality Software

Quality software development comes from discipline in following an established methodology. Top down structured programming can produce software that is (8:62):

1 correct (hard to determine);

2 reliable (no detected errors);

3 valid (meets specifications and is suitable for the job);

4 resilient (degrades gracefully when things go wrong, checks for errors, and provides recovery routines);

5 usable (shows consideration for human factors consistent conventions, few if any arbitrary codes/names, through diagnostics and error messages);

6 clear (design structure apparent from program listing, meaningful names, use of well known algorithms, frequent and effective comments, modular structure);

7 maintainable/modifiable (a by-product of clarity, changes due to detected errors or system changes are easily incorporated);

8 generalized (performs over a wide range of input values, modes, and use);

9 portable (hardware specific and software dependent features are isolated for easy change to another computer system);

10 and is testable (step by step testing is possible due to simple structured algorithms).

Programming Guidelines

The following guidelines for structured programming provide a methodology for program development (8:69):

1 Program in small modules.

2 Comment programs telling what and why things are done and what assumptions exist if any.

3 Don't misuse the instruction set or software language.

4 Don't write self modifying programs.

5 Avoid complex statements - break them up into smaller parts.

6 Use indentation and a format that makes listings more readable.

7 Avoid negative Boolean logic. Reversing an if - then clause allows dropping a NOT in front of an expression.

8 Use meaningful names for variables, constants, and procedures.

9 Make modules that do not interfere with the code or data of other modules.

10 Uncommented code that works is better than commented code that doesn't work, until it comes time to modify the code. Comments should clarify. Clarity is its own reward.

Decision Making

A major part of a robot control system is decision making. The more decisions that can be deferred until run time, the better the robot program can be adapted to changing task requirements. There are four kinds of decisions that may be deferred until run time (8:74):

1 What initial data items are required.

2 How to allocate resources.

3 How to coordinateconcurrentprocesses.

4 Howtohandle exceptions.

Deferred Data Items

Determining deferred data items may require communication with an external computer system, locating an index mark or calibration jig for sensor alignment, or even human interaction with the robot computer and/or sensors.

Resource Allocation

Resource allocation becomes a problem when two or more processes require the same resource. This still may be deferred past the start of run time to the concurrent process coordinator to resolve once the resources are requested during program execution.

The concurrent process coordinator switches the attention of the central processor and possibly other system resources between processes by use of a semaphore or Dijkstra flag. In multiple processor, multiple process, and/or interrupt driven robots, the use of special hardware (or a software instruction like the afore mentioned MC68000 test and set instruction) may be required to insure resolution of resource allocation conflicts. The coordinator must never allow a situation to exist where two processes or processors have been allocated resources and will not release them to another and yet can not proceed until a resource controlled by another is obtained. This is the classic deadlock or deadly embrace.

Exception Handling

Exceptions are either predictable or unpredictable (8:77). Predictable exceptions occur when a verification step returns false such as a mobile robot not finding a position update marker as planned. Unpredictable exceptions occur during well defined procedures. These are hardware

failures or the selection of a software path that was not tested and has an error. Software failures are best treated with preventive measures earlier discussed. Hardware redundancy, parity bits, checksums, cyclic redundancy check characters (CRC), message sequence numbers, send and receive addresses, and error detection/correction codes are possible ways to decrease or handle unpredictable exceptions.

It may be advisable to disable the robot on certain exceptions. Time-outs may be implemented in hardware or software to disable a robot if a "keep alive" signal is not updated every so often. If disabled by a time-out, the robot should only be restarted by a special manual or automatic procedure. Robots should have deadman and panic switches to protect both humans and equipment.

High Level Language Programming

Programming should be done at the highest level of language possible consistent with the needs of efficiency and clarity. It is imperative that task level programming discussed earlier be done in an interpretive or haltable/restartable language so as to be able to debug programs while running on the robot. Software facilities required for program debugging include utilities to:

1 Up-load/down-load code from an off-line (off-robot) software development system to the robot;

2 Relocate code in memory;

3 Link modules;

4 Allow Symbolic debugging;

5 Set/reset breakpoints; and

6 Single step program execution.

Programming languages used should conform to the structured programming guidelines. Possible languages include:

1 Assembly language in the processor's instruction set;

2 The C programming language;

3 Structured FORTRAN;

4 PASCAL;

5 ADA;

6 LISP;

7 PROLOG; and

8 FORTH.

Many other good languages exist, but the mentioned languages have virtues which make them desirable for program development and debugging. There are also many robot specific languages but are usually processor and hardware dependent. These can, however, be used as a basis for developing new robot languages if concern for quality software design is maintained.

A quality editor/word processor software package is a must for software development, documentation, and reporting.

Sensory Data Processing

The only arguments against the use of a particular sensor are its cost in terms of software overhead, required computing power support, time to process the sensor's data, fit on robot, weight, robot capability without the sensor (which may be an argument for the sensor), external equipment/personnel support required to use this sensor, and money (8:25).

The major robot sensor categories are:

1 Proprioceptors (sense position);
2 Touch;
3 Proximity;
4 Range;
5 Force;
6 Movement;
7 and Vision (really only a subset of the vision that humans experience).

Sensor Error

Sensor errors may be due to changes in sensor characteristics over time, quantization error, or sensor susceptibility to noise of some form. Filtering techniques may therefore have to be incorporated to compensate for or minimize the effects of noise.

Crowley has proposed a computational paradigm or model for three dimensional scene analysis (17). He explains that

multiple sensor systems may be used to gather information about a robot's environment and a sensor model of the environment developed. Models from many sensors are combined to create a combined or composite model that can change with a dynamic environment. This composite model is then used to solve a mobile robot's problem of global navigation, local navigation, and position estimation (18).

What is a Robot

A robot is a device that performs functions normally ascribed to human beings, operates with what appears to be almost human intelligence, or is a mechanism guided by automatic controls (19:744). Industrial robots generally consist of an arm bolted to a platform with an end effector (gripper, spot welder, drill, or special tool) affixed and which is controlled by a computer. A robot is distinguished from a Numerically Controlled (NC) machine (such as an automatic lathe) in that a robot's actions are determined by sensor feedback and not just a sequence of computer instructions. Most mobile robots still remain in research labs because of the demanding requirement to have some form of vision to allow obstacle avoidance and object identification to allow a solution to the mobile robot navigation problem. New forms of vision based on range and direction information may soon allow mobile robots to function in environments previously requiring human workers.

III. MARRS-1 Design

Overview

One of the major efforts of this thesis has been to develop hardware and software to control and communicate with the AFIT MARRS-1 robot. Additionally, three important steps were taken to improve the robot. First, the robot was taught and run thru a simple sequence of operations to identify the errors involved with moving the robot in various directions. Second, the robot was then modified to make optimum use of its sensors and to eliminate where possible the errors identified in the first step. And thirdly, a software development system was established to aid in the design, testing, and maintenance of the robot and its subsystems.

After a period of hardware evaluation, mechanical redesign, fabrication, and electronics upgradeing was completed, the remaining task involved many iterations of program development, testing, data analysis, and program modification based on test results.

A series of Robot Integrated Operation Tests (RIOTs) were then performed that have great value in many key areas. The tests establish a benchmark that can be used as a point of reference for further work. In addition, if the errors in a particular device are consistently similar to previous test cases, then error compensation is relatively simple.

If on the other hand, the errors are random, then much more emphasis must be placed on multiple sensor feedback. The tests may be expanded as the mobile robot project continues and should be updated as automated tools are made available (or developed). The objective of the tests should not only be to demonstrate a capability but to also identify deficiencies that may exist in the robot and to what extent.

Familiarization with the AFIT HERO-1 Robot

Before a prospectus for this thesis could be submitted, some familiarity with the AFIT HERO-1 robot built by Lt. Owen was necessary (see figure 3.1). What could the robot do? What would be the next step in developing a truly autonomous mobile robot? Could the robot be made autonomous? The answers to these questions would be the driving force in determining the direction of this thesis.

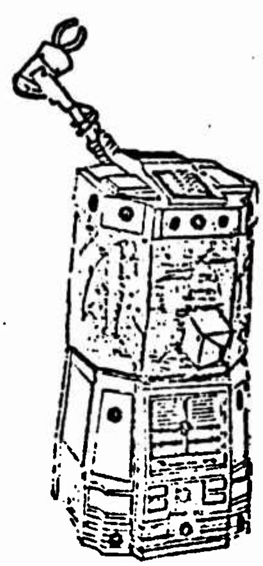


Figure 3.1 AFIT HERO-1 Robot Built by Lt. Owen

What Could the AFIT HERO-1 Robot Do?

The AFIT HERO-1 robot had an arm with 145 angular degrees of movement, a wrist with two axis of motion, and a two claw gripper. Locomotion was developed from a tricycle gear wheel configuration with the front wheel the only wheel actively propelled and steered plus or minus 90 degrees from center. The drive of the front wheel had three speeds in both the forward and reverse directions. There were two levels to the body with the top level being able to turn plus or minus 178 degrees. Additionally, the robot had a Votrax SC01 speech synthesizer, a single three degree beamwidth ultrasonic sonar range finder in its head with separate transmit and receive transducers, a calendar with clock, a 256 level light intensity sensor, a 256 level sound volume sensor, and an ultrasonic motion detector. Lt Owen had added a laser barcode reader and 15 Polaroid type ultrasonic sonar range finder transducers. The robot had an impressive list of resources for mobile robotics research, could make crude maps of its environment, and move about randomly without hitting obstacles under self control. Yet, the AFIT HERO-1 robot had some major deficiencies.

Deficiency Identification

Initial investigation into the opportunities for improving AFIT's mobile robotics research revealed the following:

The ability to rapidly develop and debug software was seen as one of the most important improvements necessary for the AFIT mobile robotics research environment.

The front wheel design of the HERO-1 robot needed to be improved since the steering gearbox bearings wore excessively over a short period of robot use. The bearing wear and the additional weight from the laser barcode reader distributed high in the robot's body caused the robot to be mechanically noncompliant, nonrepeatable, and top heavy (unstable). Motion of the robot now became random since commands to the stepper motor controlling the steering of the front wheel, were sent from the robot's computer and assumed to have been obeyed, when in fact, it was very easy for small impeding forces to withstand and overcome the torque of the steering stepper motor. Thus, robot steering and position estimation algorithms were defeated. This condition could have been avoided in part by determining the position of the steering shaft explicitly with a sensor and implementing a closed loop feedback system.

The AFIT HERO-1 robot Polaroid ranging system was not as fast in gathering data as it could have been since only one transducers could be used at a time.

The creation of a sonar range map from data gathered by the AFIT HERO-1 robot required undesirable human intervention to:

1. Determine the robot's heading for each sonar sample period.

2. Provide a robot position correction factor for each sonar reading. The barcode that was read by the robot was taped to the floor and was from one and a half to two feet away from the center of the robot. Thus, position of the robot was loosely based on the number appearing on the barcode. The position of objects in the test environment was based on data from the sonar system which was not firmly referenced to the barcode taped to the floor.

3. Draw the two dimensional (birds eye view) map post mission in a form useful to humans from sonar data, barcode position information, and notes taken by human observers during the mission (or test run).

The AFIT HERO-1 robot was then redesigned, rebuilt, and renamed the AFIT Mobile Autonomous Robot Research System (MARRS-1) and is shown in figure 3.2.

System Level Description

The overall system structure for control of MARRS-1 is shown in figure 3.3. Note that the vertically integrated control structure from the external computer to the Nav Computer to the Drive computer is only one of many configurations possible. The RIOTs of chapter 5 used this structure minus the control shown from the Nav Computer to the Drive Computer.

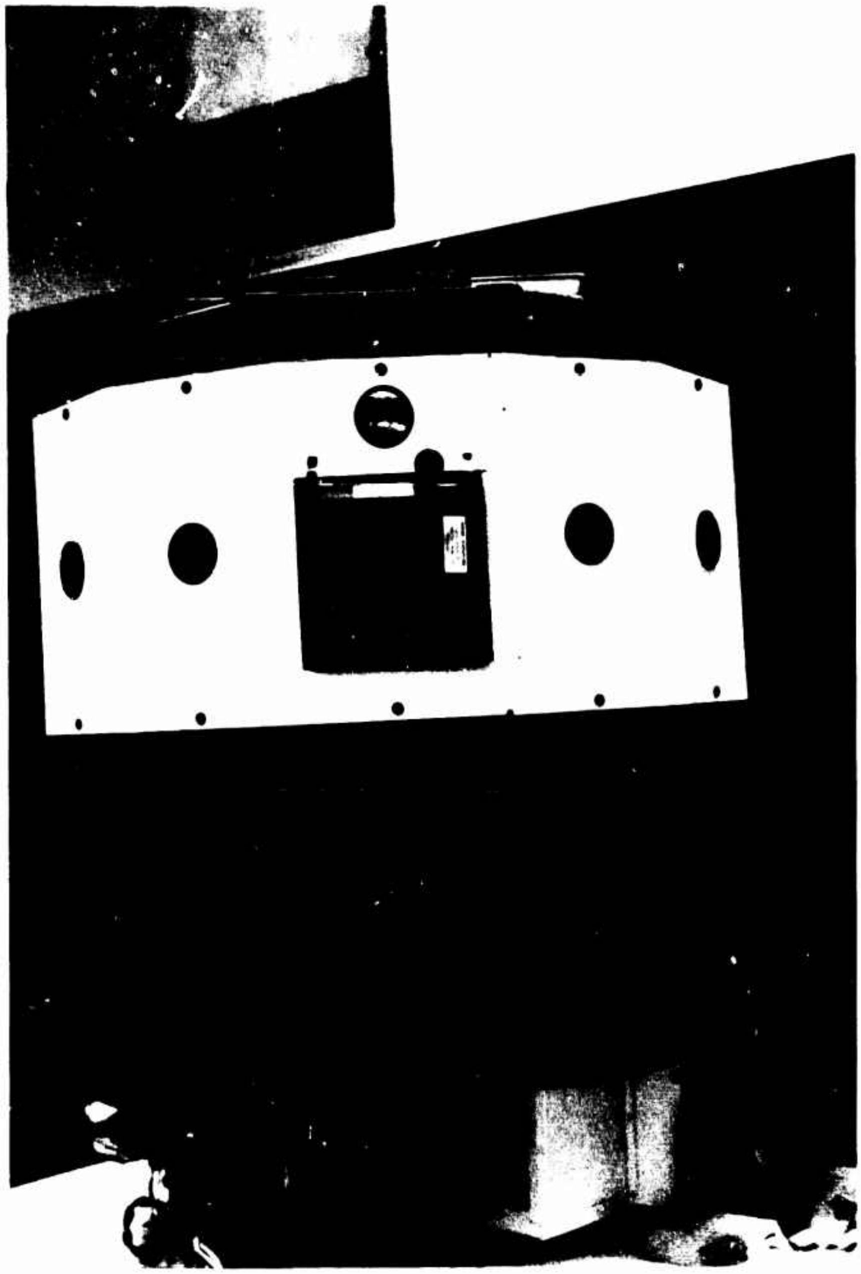


Figure 3.2 AFIT Mobile Autonomous Robot Research System (MARRS-1)

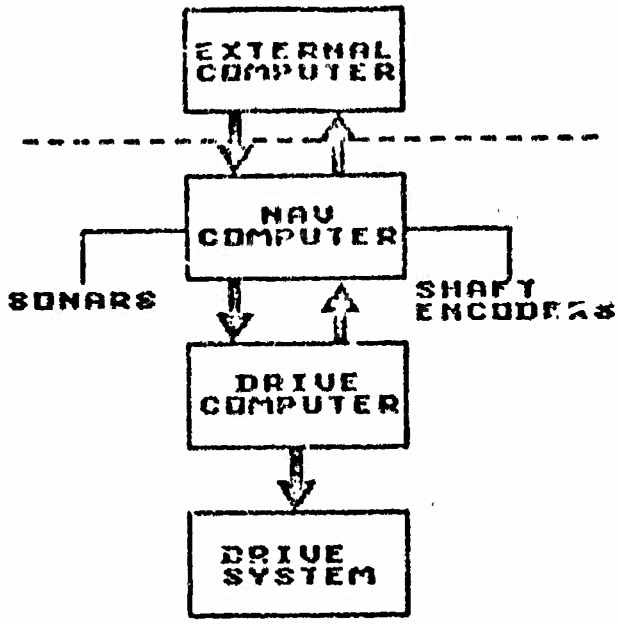


Figure 3.3 MARRS-1 Control Structure

Design Philosophy

Several key factors have molded the design of MARRS-1 into its final form. The first and foremost important design criteria was: Never discard a capability, only add to or enhance the original design. The second design criteria was to treat all system components and sub-elements as black boxes with standard interfaces (in this case serial RS-232). Modularity and expansion was always kept in mind as systems were designed and developed. Hardware implementation of a given task was always chosen over software (if a trade-off existed) to alleviate unnecessary burdens from the CPU.

Mechanical Design

The MARRS-1 robot is 29 and a half inches high, 21 and a half inches in diameter at its widest point, has two decks with 12 sides each, an improved tricycle configuration with a front wheel only drive system, separate shafts for each rear wheel, and a low slung 8 gel cell battery compartment. It has retained all the features of the HERO-1 robot except those modified to improve performance and the arm. The arm was intentionally left off MARRS-1 since the intent is to work toward solving mobile autonomous robot problems and not fixed arm problems. Once a good autonomous mobile platform has been developed, it will become desirable to add a quality robot arm with special end effectors. The twelve side design grew out of an attempt to have symmetrical sonar coverage at two levels erroneously thinking that the beamwidth of the Polaroid sonar transducers was 30 degrees when they are more like 10 to 15 degrees. The twelve side design impact on the sonar system will be discussed later.

Drive Computer

The original HERO-1 contained a MC6808 computer system with no RS-232 serial capabilities. The Virtual Devices Inc. MENOS-1 MC6801 upgrade CPU board enhanced the HERO-1 and gave it RS-232 capability (at 300 baud however) and tremendous new programming possibilities. The MENOS-1 ROM was modified to support 9600 baud rate communications

(see Appendix E). It is now possible to program the Drive Computer in the C language, in Virtual Devices tokenized Vamp language, in HERO-1 learn mode (teach pendant), in HERO-1 Robot language, and in 6801 Assembly language (a superset of the 6800 family with internal 16 bit operations). The Drive Computer (Menos I upgraded HERO-1 computer) is described in Virtual Device's Menos I user's manual, Virtual Devices Robo C user's manual, Heathkit's HERO-1 documentation, and in Appendix E.

Navigation Computer

The Navigation Computer is the keystone of MARRS-1. It controls both the Optical Shaft Encoder Subsystem and the Sonar Subsystem.

Serial RS 232 communication links are provided from the Nav Computer to both the Drive Computer and an external computer. Programs can now be cross assembled and downloaded from an external computer via serial ports to either the Nav or Drive computers on the MARRS-1 robot. In addition a third serial interface connects the Nav Computer to a laser 3 of 9 barcode reader enabling MARRS-1 to run software developed by Lt Owen. However, the laser barcode reader was not used for the Robot Integrated Operation Tests (RIOTs) (chapters IV and V).

Nav Computer Hardware

The digital portion of the Nav Computer is centered around Motorola 6800 family devices (see Appendices F and G). The operating system fits entirely on an 8K ROM with room to spare. An additional 8K ROM is located in parallel with the base page 8K RAM. System initialization software and page select circuitry load into the base page 8K RAM the contents of the base page 8K ROM after any hardware reset or call of the subroutine labeled ROMLAYRAM. A full contiguous 48K of static RAM exists on board. Input and output devices are memory mapped as is the case with all Motorola 68XX, 68XXX devices.

Nav Computer Software

MARRSBUG (see Appendix A and D) is an interrupt driven, heavily modified serial version of Motorola's MIKBUG and American Microsystem's PROTO operating systems. Data from the Nav Computer's Sonar and Optical Shaft Encoder subsystems is constantly being updated by a maskable hardware interrupt handler and made available to any user programs that run on the Navigation Computer. Hence, the net effect of the MARRSBUG interrupt driven data acquisition system is transparent to the user. The operating system is flexible enough to allow changes to vectors and constants which are loaded into system RAM on power up. An important feature of MARRSBUG is that on power up, the system performs a RAM check to verify where

the largest contiguous block of good RAM is located and updates the system RAM vectors appropriately. Additionally, numerous callable subroutines, software interrupts, and system utilities exist within the ROMs to aid the development of application programs for the Nav Computer (see Appendices A,B,D, and N).

Optical Shaft Encoder (OSE) Subsystem

The OSE Subsystem is perhaps the most important of the two Nav Computer subsystems (although one subsystem cannot perform to its utmost without the other). A 1200 count per revolution OSE is placed on the shaft of each rear wheel (two independent shafts) and on the front wheel steering shaft. By maintaining the distance traveled by each rear wheel, both heading and position information (relative to initial heading and position) are readily obtainable by calculating simple trigonometric equations (see Chapter V). The instantaneous steering position of the front wheel is maintained by subtracting the clockwise and counterclockwise counts of the front wheel steering OSE.

Cumulative counts are required to integrate this sensor data into the Navigation Computer. Thus, an incremental encoder was chosen. The Datametrics K3 encoder (see Appendix K) provides both incremental pulses and two channels which are 90 degrees out of phase with each other. With the proper interface circuitry,

this phase relationship enables the robot to distinguish between forward and reverse motion of each rear wheel.

A Motorola 6840 Programable Timer/Counter chip, in conjunction with a DM9602 dual precision one shot multivibrator, provides an optimum circuit to interface each OSE with the CPU (see Appendix G, Figure G.15). By using a 6840 as a counter (of which there are three 16 bit down counters on each chip) the burden of keeping track of continuously rotating wheels is removed from the CPU.

The DM9602's determine wheel direction (forwards or backwards). This is accomplished by testing for a low signal on one OSE channel and a negative transition on the other OSE channel. The 6840's act as divide by 64 counters which equates to one inch of wheel travel. When this distance has been reached, an interrupt is generated and a variable counter in system RAM is incremented (cumulative counts). Since the 6840 is programable, this divide by XX value may be modified to suit the user's scale factor and precision needs (see chapters V and VI).

Ultrasonic Sonar Subsystem

The AFIT MARRS-1 robot utilizes a simple vision system to aid in the solution of the mobile robot point to point navigation problem. The system acquires range and direction information from 32 polaroid sonar transducers attached to the robot's exterior.

Figure 3.4 illustrates the placement of 24 of these transducers on the robot as seen from an overhead view.

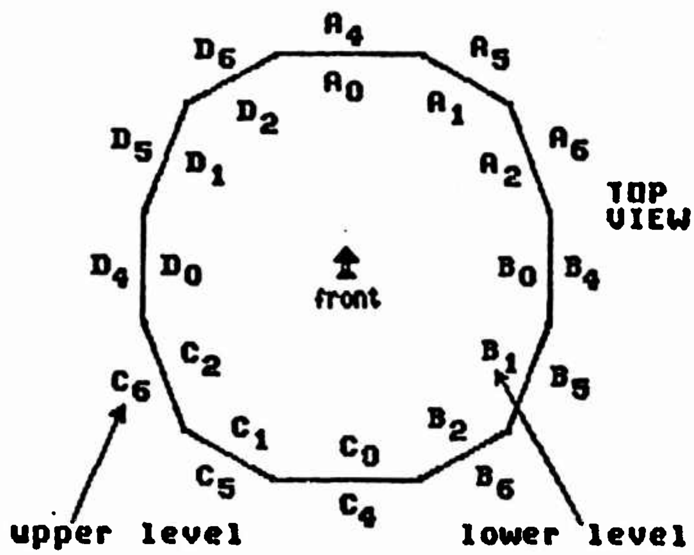


Figure 3.4 Sonar Transducer Locations

The robot's sonar sensors are divided into two different groupings by physical location on the robot or into four different groupings according to their attachment to one of four sonar range boards (Group A, B, C, or D). The lower body deck houses sonar transducers labeled AØ, A1, A2, BØ, B1, B2, CØ, C1, C2, DØ, D1, and D2. The upper body deck is comprised of sonar transducers labeled A4, A5, A6, B4, B5, B6, C4, C5, C6, D4, D5, and D6. There are an additional eight transducers in the head of the robot but

are not used in the RIOTs (Chapter V, Chapter VI). The selection of these eight transducers is however supported in the MARRS-1 operating system. The lower and upper body decks have 12 equal sides. Each side points 30 degrees away from its two neighbors on the same deck. The drive computer can rotate the upper deck plus or minus 178 degrees by controlling a stepper motor. Each sonar transducer is electrically connected to one of four sonar range boards (Sonar Board A, Sonar Board B, Sonar Board C, and Sonar Board D) by means of coaxial cable to one of 32 dual inline package (DIP) relays. Transducer selection is accomplished by energizing a DIP relay.

Only a maximum of four sonar readings are valid during a sonar sample period (one from each sonar range board). If multiple paths exist between a range board and its associated transducers then direction information is forfeited and maximum detection range is attenuated. It may be useful however to energize more than one transducer at a time and establish a minimum distance for obstacle avoidance. In this mode of operation, a full 360 degree coverage can be obtained each sample period if the user is willing to disregard the direction information. Distance information can be obtained from four mutually exclusive quadrants with one quadrant covered by each range board by turning MARRS-1's upper deck 15 degrees clockwise.

Scan patterns may be established to give discrete 360 degree coverage (with individual readings corresponding to each 15 degree segment) by selecting only one transducer of six attached to a range board. Range readings are taken from one, two, three, or four range boards during a sample period and then a different combination of transducers are selected during subsequent sample periods until receiving range readings from all 24 transducers.

By carefully choosing the transducers selected, separations of 90, 120, or 180 degrees between selected transducers may be achieved during each sample period when four, three, or two range boards are used during the sample period respectively so as to minimize the possibility of intersonar interference.

Software Development Support

An H89 running CP/M was chosen to do the cross assembly and program development due to the large amount of software supported by it and the abundance of H89 and CP/M systems at AFIT. A number of cross assemblers (running under CP/M) were found for the 6800 family of processors. Two of the cross assemblers were even in the Public Domain. In contrast, it was difficult to even find a 6800 based system to do software development work on for the existing HERO-1 computer let alone the software tools such as assemblers, compilers, and editors to do the job.

Most software development efforts for MARRS-1 took advantage of Virtual Devices Robo C compiler and Robo Assembler.

The next best alternative to the nonexistent universal robot programming language was to develop a library of robot software routines. These are generic in nature so that they may be combined to form larger modules of software. A MARRS-1 Nav Computer user manual is provided in Appendix N. Software provided by Virtual Devices for Menos I, Robo C, and Robo Assembler contain many excellent routines for controlling the original HERO-1 resources and the 6801 upgraded Drive Computer. However, autonomous operation of MARRS-1 will require integrated control of the Nav Computer, Drive Computer, subsystems, resources, and possibly sensors and computers yet to be developed.

IV. MARRS-1 System Integration

Once the MARRS-1 design was completed, a number of projects were tackled in order to bring the robot to a point where it could be tested as a system. Specifically:

1. The new two deck, twelve sided body, battery compartment, reinforced front wheel, and rear wheel assemblies were fabricated by the AFIT Shop Personnel.

2. The robot was painted red white and blue so as to create interest in the goals of the thesis and for patriotic impact.

3. All electrical wiring for the original HERO-1 was replaced with longer cables since the electronics were now mounted on swing out doors to allow ease of access for modification or maintenance.

4. Operation of the robot with the original HERO-1 electronics and new wiring was verified before installing the MENOS-1 6801 upgrade computer. It became evident that the modification to the front wheel (see figure 4.1) had made the robot's movements repeatable and compliant to program command. On six different test runs the robot was able to return to the same spot (within three inches) after going over a 50 foot figure eight course in the HERO-1 learn mode.

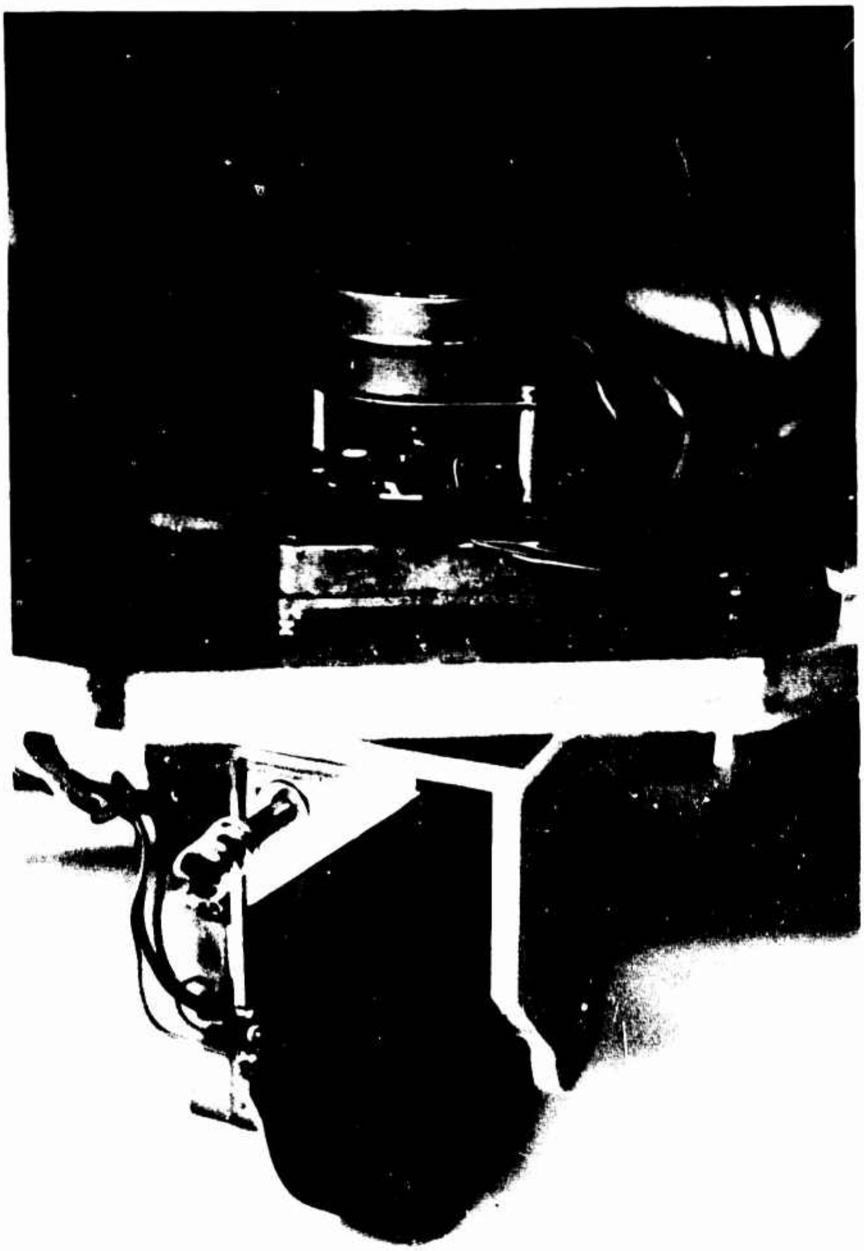


Figure 4.1 Modified Front Wheel Assembly

5. The MENOS-1 upgrade was then installed and tested. Proper operation of the original HERO-1 electronics and the new serial communication port was verified by connecting the robot to the lab H89 computer. The resulting combination of robot electronics was named the Drive Computer so as to distinguish it from the electronics yet to be added. The Input/Output (I/O) decoding scheme for the Drive Computer was extended (as shown in Appendix H) to allow further I/O expansion.

6. The Optical Shaft Encoder (OSE) Subsystem was built, installed and initially connected to the Drive Computer thru a 40 pin connector. In spite of extreme efforts to be careful, four glass encoder disks were cracked or broken while trying to install the OSEs (Appendix K) on the front wheel steering shaft (see figure 4.1) and the rear wheel shafts. The alignment of the OSEs took on the order of four hours for each encoder. What is needed is an encoder that that is sealed, prealigned, small, inexpensive, and connects to the shaft with a flexible linkage.

7. The Nav Computer was wirewrapped onan Augat prototyping board and hardware/software debugged using a Hewlett Packard 1610A logic analyzer. The OSE Subsystem was disconnected from the Drive Computer and connected instead to the Nav Computer with the Sonar Subsystem. The two subsystems conceptually can be used as a crosscheck against each other if a priori knowledge about an environment is

given and function optimally if the subsystems are connected to the same virtual computer.

8. The Sonar Subsystem was connected and debugged on the Nav Computer only after proper operation of the computer (CPU, ROM, RAM, and interrupt driven I/O devices) was verified.

Sideline Activities

Many side issues, projects, and activities consumed time during the thesis and in their own way contributed to the success of the effort. These included:

1. Participation in the 10th Annual Dayton International Air Show representing AFIT with the newly painted and rewired MARRS-1 robot.

2. Taking MARRS-1 to the National Explorer Scout Fly-In at Columbus, Ohio (once again to represent AFIT).

3. Television, radio, and newspaper interviews, along with presentations to public school groups, kept MARRS-1 (and its creators) busy informing the public about AFIT research.

4. The AFIT Mobile Robotics Laboratory was created during the thesis by scrounging for resources necessary to do this type of hardware/software development project. The physical space of the lab was absolutely essential for the success of the thesis. The floor of the test area had to be leveled by using wall plaster to fill in holes and depressions.

5. Two other computer systems were built during the thesis effort to support software and hardware development. A Motorola Exorciser and a 6802 based printer controller card (previously developed for an AFIT project) were the only 6800 family computers in the school. Both needed extensive modification and rehabilitation to meet the development needs of the thesis.

With the robot hardware and software development complete, a series of Robot Integrated Operation Tests (RIOTs) were conducted.

Robot Integrated Operation Test (RIOT) Plan

The RIOTs demonstrate the ability of MARRS-1 to keep track of its own location and map out the environment. The RIOTs also identify anomalies (if any) in robot and algorithm operation while they (the tests) manipulate data collected from both the OSE and Sonar subsystems into a form useful for robot obstacle avoidance, path planning, and robot performance benchmarking.

Test Range Environment

The test range used for the RIOTs is part of the AFIT Mobile Robotics Laboratory with dimensions of 13 feet by 25 feet. All doors to the test range remain closed during each test run. Test conditions are broken into two categories; tests without objects in the test range and tests with two objects in the test range (see figures 5.1 and 5.2).

Object number one is composed of a large 3' diameter 2' 7" high cylinder. Object number two is a box with the following dimensions: 1' 6.5" wide, 2' 9.5" long, 2' 3" high. Both objects have cardboard material surfaces. The test range floor is a reasonably flat tile surface. The walls of the test range are painted hardboard. Some minor protrusions such as door knobs, conduit, and molding jut out from the surface of the walls. All walls are assumed to be flat surfaces since the Ø.1 foot sonar range resolution of cannot accurately distinguish these protrusions.

MARRS-1 Operating Modes

In order to collect enough data to form a map of the test range the robot is operated in an open loop mode and moved throughout the test range. Two patterns of movement will be tested; a straight line path and a zig-zag pattern. The initial test run location and heading of the robot should be kept the same (in this case X = 2 feet, Y = 7 feet) for ease of setup. Sight tubes, mounted on the robot, aide in the initial positioning of the robot. The upper level head of the robot is rotated 15 degrees clockwise to provide a 36Ø degree coverage from the 24 ultrasonics transducers with 15 degrees of separation between each transducer. Additionally, the number of sonar transducers selectedduring each cycle of the interrupt handler varied from test to test. Table 4.1 lists sixteen different test configurations that were used.

TEST #	OBJECTS	PATH	# SONARS SELECTED PER PERIOD
4	TWO	STRAIGHT	2 PER .1 sec
5	TWO	STRAIGHT	1 PER .1 sec
7	TWO	STRAIGHT	3 PER .1 sec
9	TWO	STRAIGHT	4 PER .1 sec
10	TWO	ZIG ZAG	1 PER .1 sec
11	TWO	ZIG ZAG	2 PER .1 sec
12	TWO	ZIG ZAG	3 PER .1 sec
13	TWO	ZIG ZAG	4 PER .1 sec
14	NONE	ZIG ZAG	1 PER .1 sec
15	NONE	ZIG ZAG	2 PER .1 sec
16	NONE	ZIG ZAG	3 PER .1 sec
17	NONE	ZIG ZAG	4 PER .1 sec
18	NONE	STRAIGHT	1 PER .1 sec
19	NONE	STRAIGHT	2 PER .1 sec
20	NONE	STRAIGHT	3 PER .1 sec
21	NONE	STRAIGHT	4 PER .1 sec

Table 4.1 RIOT Configurations

Support Equipment

The following equipment provides the necessary support to conduct this test; the AFIT MARRS-1 robot, Heath H-89 computer with H-27 8 inch disk drive system, Modem 720 communication program, and an RS-232 cable. Post mission processing of the collected data is performed on a TRS-80 (6809 based) Color Computer.

Drive Computer Learn Mode Programs

The learn mode programs shown in tables 4.2 and 4.3 run in the Drive Computer and cause the MARRS-1 robot to perform either straight line motion or a zig-zag pattern within the confines of the test range. (For a more detailed discussion ofrobot interpreter commands the reader is referred to the Heathkit Educational Systems ET-18 ROBOT Technical Manual.)

ADDRESS	DATA	HERO-1 ROBOT LANGUAGE AND MEANING
Ø4ØØ	C3 19 6F	MOTOR MOVE, WAIT ;ABSOLUTE(IMMEDIATE) DRIVE MOTOR SELECTED FORWARD $16F FAST GEAR
Ø4Ø3	8F ØØ 2Ø	PAUSE (IMMEDIATE)
Ø4Ø6	3A	RETURN TO EXECUTIVE ("READY")

Table 4.2 MARRS-1 Straight Line Drive Computer Program

ADDRESS	DATA	HERC-1 ROBOT LANGUAGE AND MEANING
Ø4ØØ	C3 18 Ø6	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $Ø6
Ø4Ø3	CC 18 15	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $15
Ø4Ø6	C3 E8 33	TURN LEFT TO POSTION $33
Ø4Ø9	CC 18 4E	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $4E
Ø4ØC	C3 E8 49	TURN STRAIGHT AHEAD
Ø4ØF	CC 18 1C	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $1C
Ø422	C3 E8 64	TURN RIGHT TO POSITION $64
Ø425	CC 18 22	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $22
Ø428	C3 E8 49	TURN STRAIGHT AHEAD
Ø42B	CC 18 12	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $12
Ø42E	C3 E8 57	TURN RIGHT TO POSITION $57
Ø431	CC 18 ØC	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $ØC
Ø434	C3 E8 49	TURN STRAIGHT AHEAD
Ø437	CC 18 ØD	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $ØD
Ø43A	C3 E8 58	TURN RIGHT TO POSITION $58
Ø43D	CC 18 39	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $39
Ø44Ø	C3 E8 49	TURN STRAIGHT AHEAD
Ø443	CC 18 17	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $17
Ø449	C3 E8 31	TURN LEFT TO POSTION $31
Ø44C	CC 18 18	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $18
Ø44F	C3 E8 46	TURN RIGHT TO POSITION $46
Ø452	CC 18 ØØ	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $ØØ
Ø455	C3 E8 42	TURN LEFT TO POSTION $42
Ø458	CC 18 25	MOVE DRIVE MOTOR (HIGH SPEED) FORWARD $25
Ø45B	C3 E8 49	TURN STRAIGHT AHEAD
Ø45E	8F ØØ FF	PAUSE
Ø461	3A	RETURN TO EXECUTIVE ("READY")

Table 4.3 MARRS-1 Zig-Zag Drive Computer Program

Robot Test Procedures

1. Objects (if any) are placed in the test range with their location and orientation documented.

2. Document and mark on the floor the initial heading and position of the robot. Heading is normalized for the program MAPPER (Appendix C) as a value between Ø and 1 as shown in figure 4.2. The (X,Y) location of the robot is represented in terms of tenth's of feet (resolution of 1.2"). The initial location and all subsequent locations of the robot are relative to the mid point between the two rear wheels. Both heading and location are crucial in conducting this test with any accuracy/repeatability and are required for post mission processing of the Sonar and OSE data.

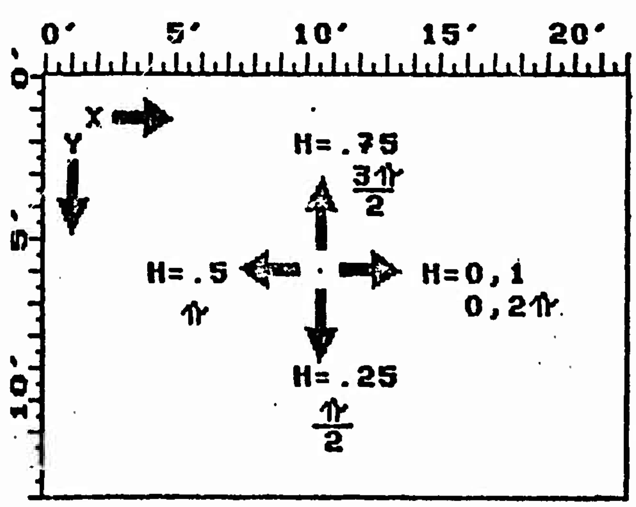


Figure 4.2 MAPPER Heading Convention

3. Straighten out the front wheel and insure that the count at memory location $0026 of the Drive Computer contains the value $49. ($49 is the value for straight forward steering.)

4. Insure that the top deck is rotated clockwise (looking down on robot) by 15 degrees and secured in place.

5. Put MARRS-1 in the learn mode and maneuver it throughout the test range. NOTE: The Drive Computer will operate the robot in an open loop mode for the entire test, i.e. there will be no feedback from the Nav Computer to the Drive Computer as a result of MARRS-1's movements or the environment.

6. After completion of the learn mode, reset the Drive Computer and bring MARRS-1 back to the marked starting position. Insure that the front wheel is straight and location $0026 contains $49. With all of this completed, execute the learned program to confirm that the robot performs adequately. If so, continue. If the robot fails to repeat the learned operation, then go back to step 2 and begin over again.

7. As in step 6, position the robot at the starting point. Now, upload the learned Drive Computer program to an external computer via the serial RS-232 port and save.

8. Before executing the test program, open a file buffer on an external computer connected to the Nav computer.

9. Reset the timer on Navigation Computer to zero.

10. Initialize the Extended Interrupt Handler to select the number of sonars activated per sample period (i.e. one, two, three, or four sonars selected during a tenth of a second interval) and execute the Drive Computer's learned test program (Table 4.2 or 4.3).

11. After each test run, insure that the captured test data is saved on magnetic media in ASCII format before continuing on with the next test run.

Data Collection

Data from each of the subsystems will be gathered every 0.1second via the Extended Interrupt Handler (see Appendix B) and stored into a temporary line buffer in the Navigation Computer's memory as follows:

/time/fw/lw1/lw2/rw1/rw2/A#____/B#____/C#____/D#____/<CR><LF>

Where time = tenth's of seconds count - 2 bytes

fw = front wheel direction - 1 byte

lw1 = left wheel reverse counts - 2 bytes

lw2 = left wheel forward counts - 2 bytes

rw1 = right wheel reverse counts - 2 bytes

rw2 = right wheel forward counts - 2 bytes

Table 4.4 RIOT Raw Data Format

The buffer is then transmitted to an external computer at 9600 baud (in this case an H-89 running M720 communication software). Buffer contents are then stored onto magnetic media for post mission processing.

Test Data Usage

After completion of the entire test all data is then transferred fromthe H-89 to the Color Computer. Using MAPPER, a mapping algorithm, (see Appendix C) the test data will be transformed into a bit plane graphics representation of the test range. Optical shaft encoder readings determine position and heading of the robot relative to the starting position (calculated with respect to the center between the rear wheels). Sonar transducer readings provide range and direction information about the robots environment from the current robot position (with respect to the center of the robot). This graphics display provides a good first order approximation of the robots environment.

V. MARRS-1Integrated Operation Test Results

Minor Test Problems

Threeminor problems occurred during the MARRS-1 Robot Integrated Operation Tests (RIOTs). However, these three anomalies did not remove the essence of the tests. They are minor and can be overcome. First, the front wheel optical shaft encoder resolution was much greater than was required. The slightest hint of mechanical vibration caused excessive jitter in the signal detected by the photo head assembly. This resulted in erroneous accumulation of counts for front wheel direction. Fortunately MARRS-1 was operated in an open loop mode. The mapping algorithm, MAPPER did not require the data from the front wheel shaft encoder. An example of these erroneous readings can be seen in Appendix O.

The second problem was with the right wheel optical shaft encoder. In spite of efforts to align the glass encoder disk, some reverse direction counts were detected while going forward. This error was of no consequence since all test runs were designed for forward motion of the robot and the MAPPER program disregarded the reverse counts from the left and right wheels. However, Navigation algorithms in general should take into account any reverse counts of either the left or right wheels. For instance, a very tight turn may cause the wheel on the inside of the turn to move

in the reverse direction. The slightly eccentric alignment of the right wheel optical shaft encoder causes erroneous reverse direction counts which may require the construction and installation of a new shaft.

The third minor setback to the completion and final testing of MARRS-1 was the inadequate current capabilities of the sonar power supply. The current surge requirements of more than one sonar transducer activated at once increased the effective internal impedance of the batteries. This reduced the output voltage applied to the sonar range boards (as well as all other subsystems) and hence maximum sonar detection range was reduced to 3.5 feet. An AC to DC power supply was employed to provide additional current drive for the sonar subsystem thus extending the maximum reliable sonar detection range to 7.5 feet.

Mapping Algorithm

The program MAPPER (see Appendix C) takes large amounts of test data and compacts it into an array of 256 by 192 picture elements (pixels) that can be displayed as a map for humans and/or used by a computer for robot control. If the heading of the robot is known at some initial starting point, then all other robot headings can be determined from the counts of the left and right optical shaft encoders. Instantaneous heading calculations may be performed if the initial heading and distances traveled by

each rear wheel are known. The following equation describes this principle (20:163):

$$HB = HA + (L - R)/D \quad \text{rad} \qquad (5.1)$$

Where HB is the current heading of the robot in radians, HA is some initial heading in radians, L and R are absolute distances traversed by each wheel starting from the initial heading position, and D is the distance between the left wheel and the right wheel. It is interesting to point out however, that the initial heading is not absolutely required since all successive heading computations are relative to the first and so on. Naturally with a digital device, a continuous sampling is impossible. Thus the traversed distance of each wheel is broken up into incremental segments. The K-3 shaft encoder has a resolution of 1200 counts per revolution. MARRS-1's rear wheel circumference is equal to 18.75". The units used in the test were inches, thus it turned out that 64 counts on the shaft encoder equalled one inch of travel by the wheel exactly. The effective resolution of the wheels now becomes 18.75 counts per revolution. Just how much resolution do the robot's wheels need for navigation and environmental mapping? These test results show that fine detailed resolution may not be required. However, optimum wheel count resolution has not been considered here and is left as an area for further investigation.

Graphics Plots

The plots shown on the next several pages are the final output from MAPPER. There are perhaps several ways of displaying the data collected from each test run. Figure 5.1 shows a layout of the size of the test range in units of feet. It was in this benign environment that half of the test runs were conducted. Figure 5.2 identifies the location of the two objects used during the other half. As a comparison, figures 5.3 and 5.4 are taken from Lt Owen's thesis and show how he accomplished mapping from a HERO-1(2:VII-6,VII-7). A key point about Owen's work is that precise heading and position information was not available for displaying the relative position of sonar readings. The remaining graphic plots (figures 5.5 thru 5.2Ø) are of the Robot Integrated Operation Tests (RIOTs) and demonstrate that a robot can create a low resolution map of its environment (relative to itself) using ultrasonic sonar as a "vision" device while keeping track of its orientation and location from wheel counts. Each sonar reading is displayed as an arc of 2Ø degrees at the distance measured from the center of the robot and plotted based on position and heading information determined from rear wheel optical shaft encoder data. The tests are termed integrated since robot performance is evaluated based on all systems working in concert.

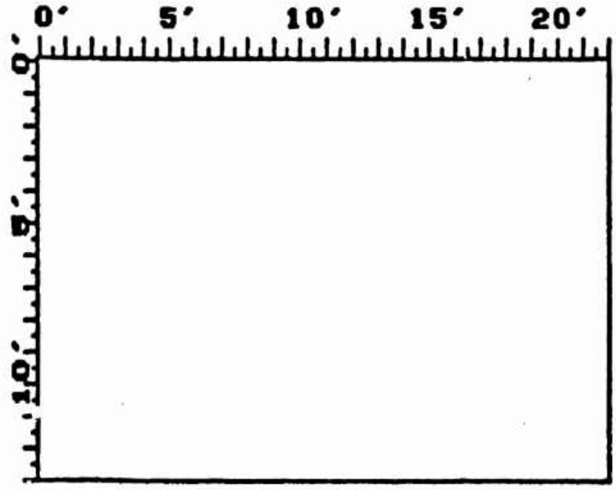


Figure 5.1 Robot Integrated Operation Test (RIOT) Range Layout

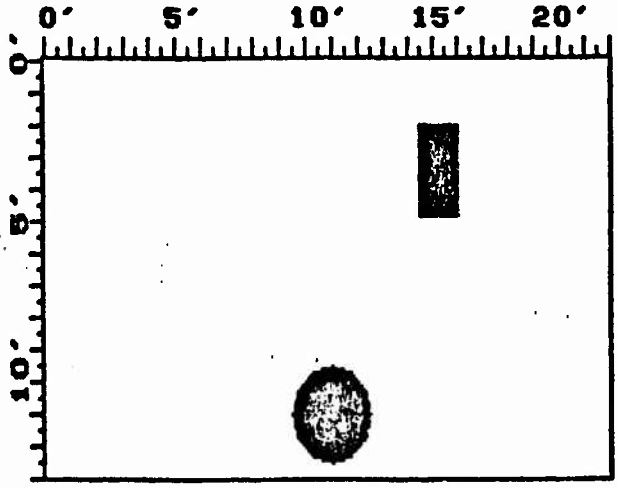


Figure 5.2 Location of Objects for RIOTs with Objects

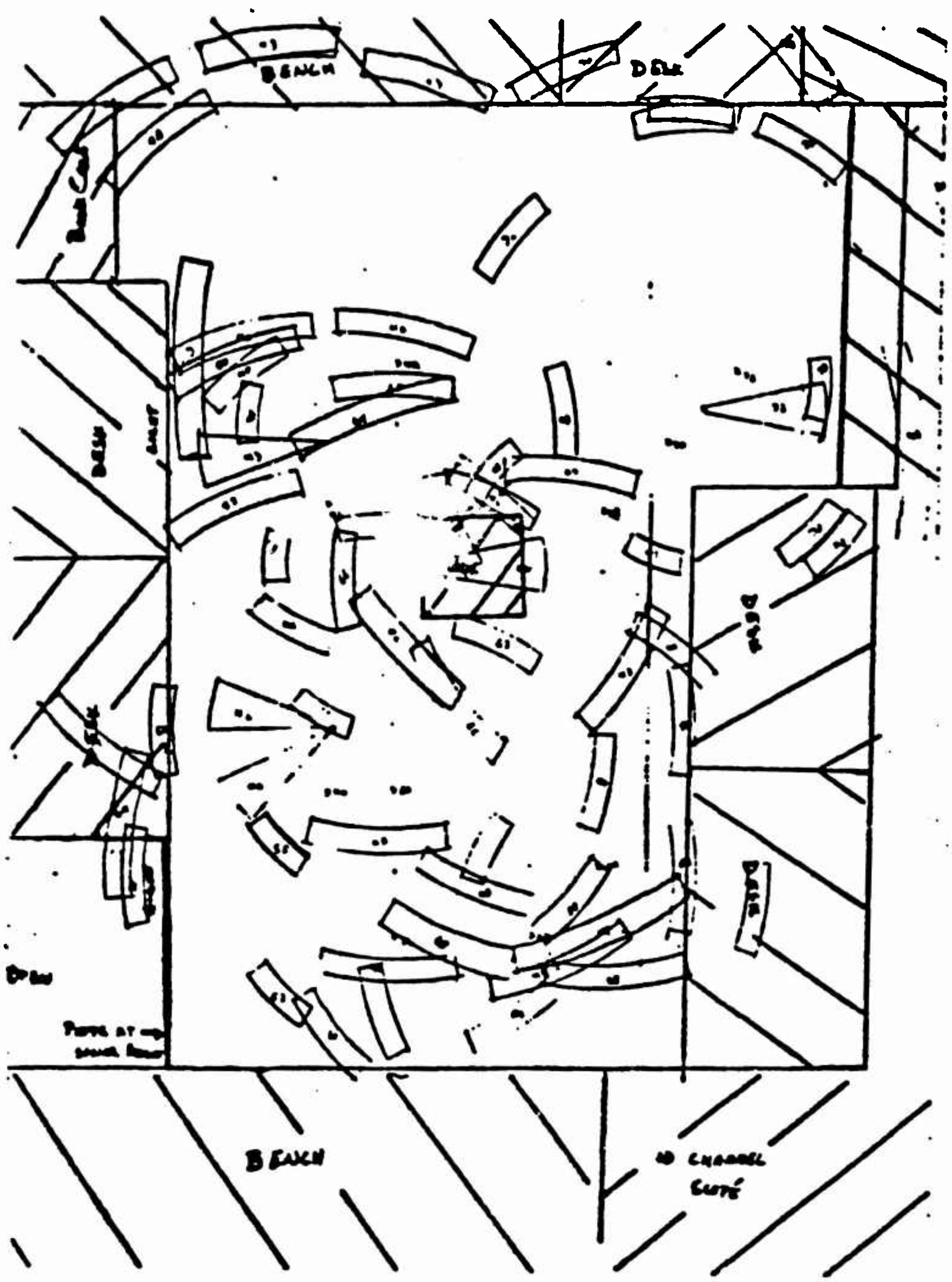


Figure 5.3   Owen Thesis Test Range Layout and Sonar Map

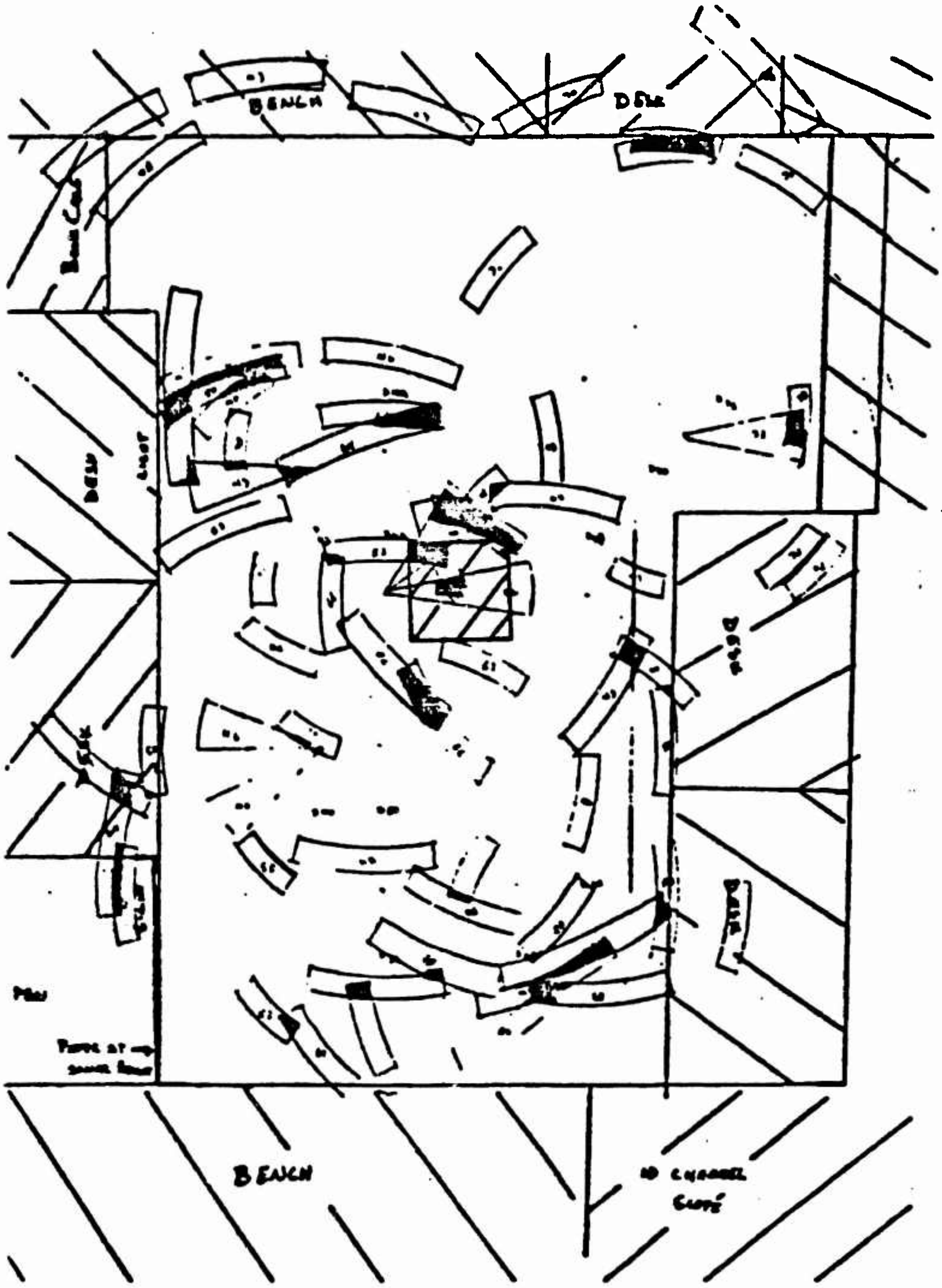


Figure 5.4   Owen Thesis Sonar Map Showing Sonar Coincidence

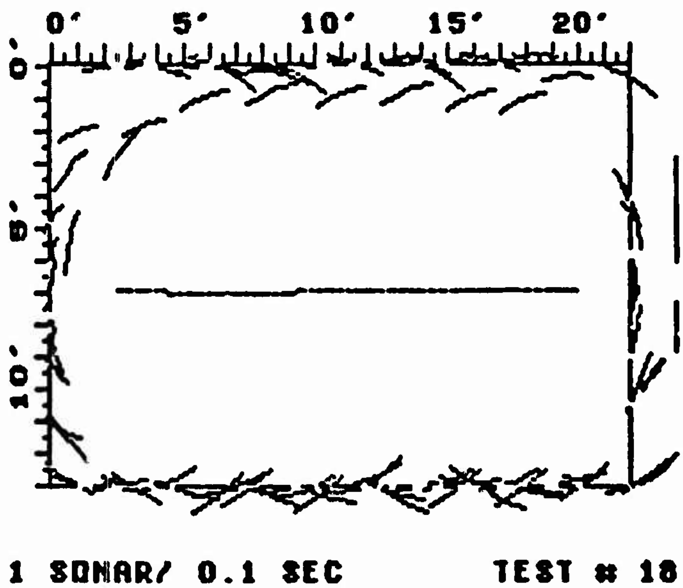


Figure 5.5 RIOT #18: Straight One Sonar at a Time

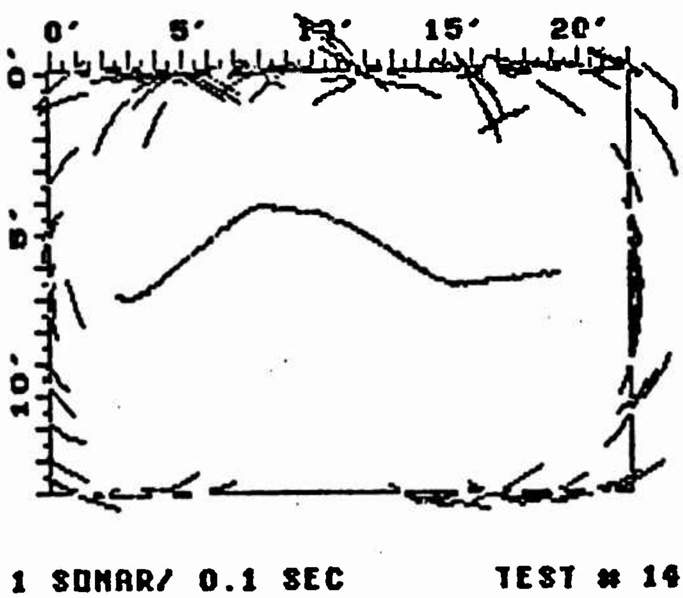


Figure 5.6 RIOT #14: Zig Zag One Sonar at a Time

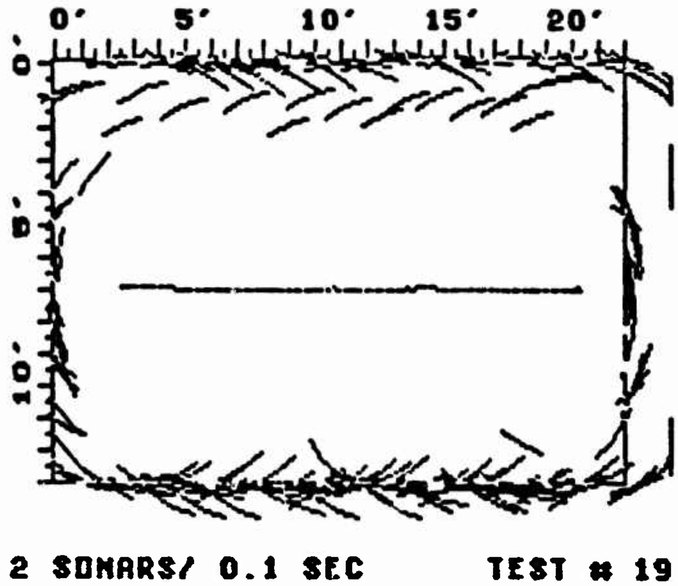


Figure 5.7 RIOT #19: Straight Two Sonars at a Time

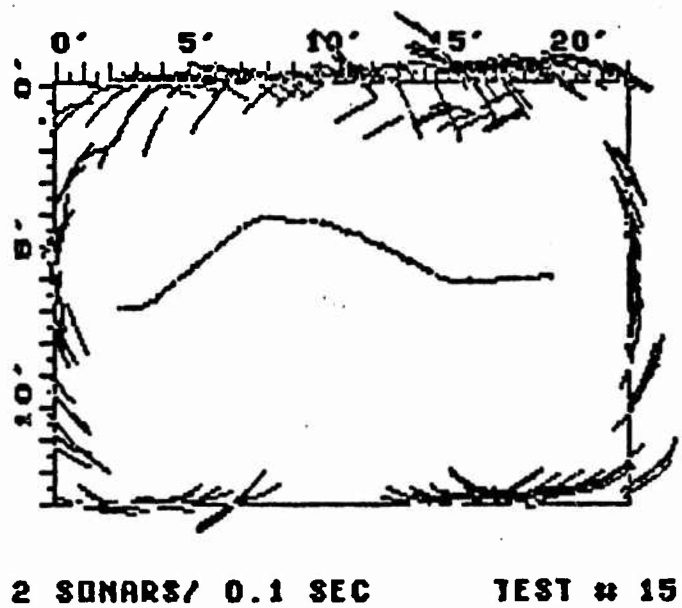


Figure 5.8 RIOT #15: Zig Zag Two Sonars at a Time

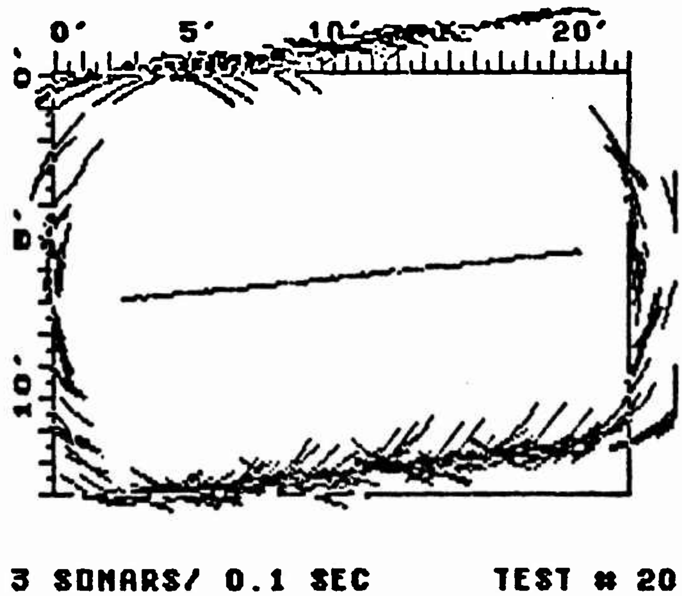


Figure 5.9 RIOT #20: Straight Three Sonars at a Time

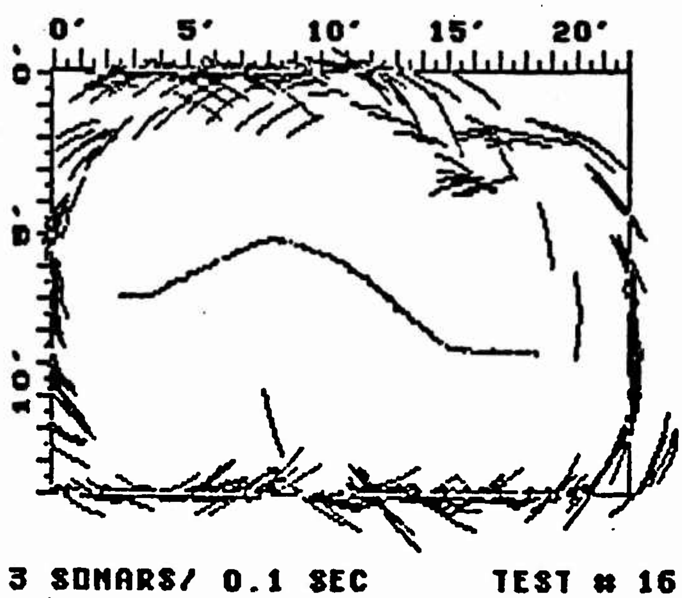


Figure 5.10 RIOT #16: Zig Zag Three Sonars at a Time

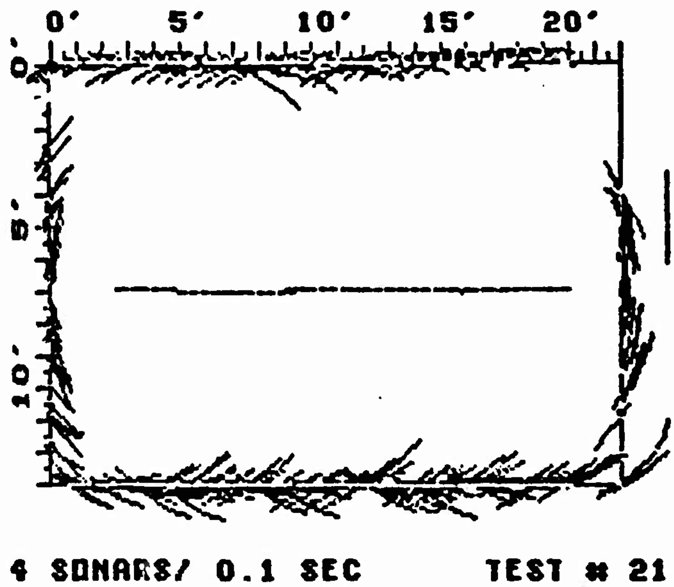


Figure 5.11 RIOT #21: Straight Four Sonars at a Time

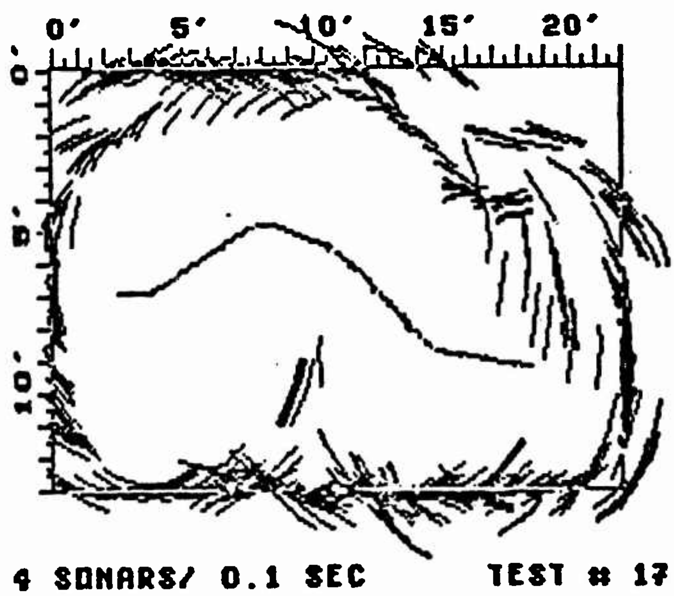


Figure 5.12 RIOT #17: Zig Zag Four Sonars at a Time

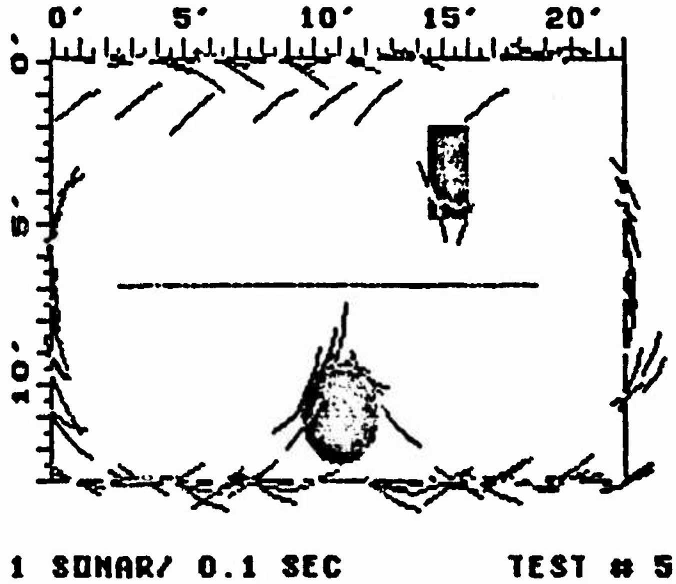


Figure 5.13   RIOT #5: Straight One Sonar at a Time

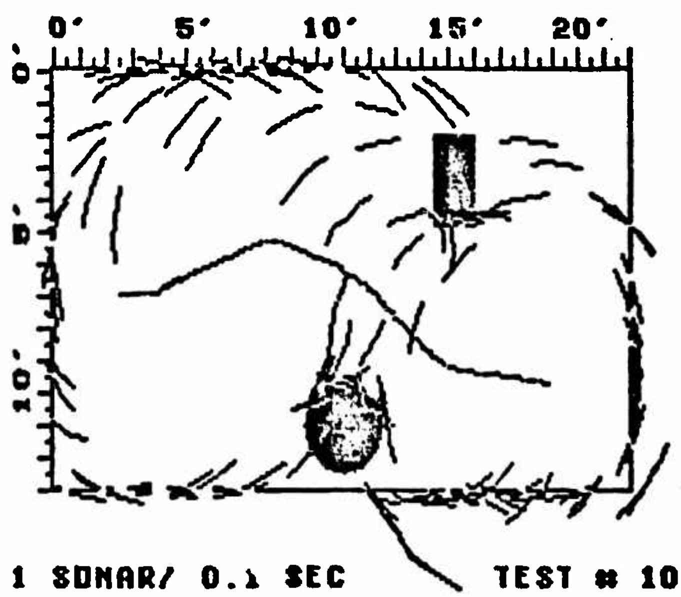


Figure 5.14   RIOT #10: Zig Zag One Sonar at a Time

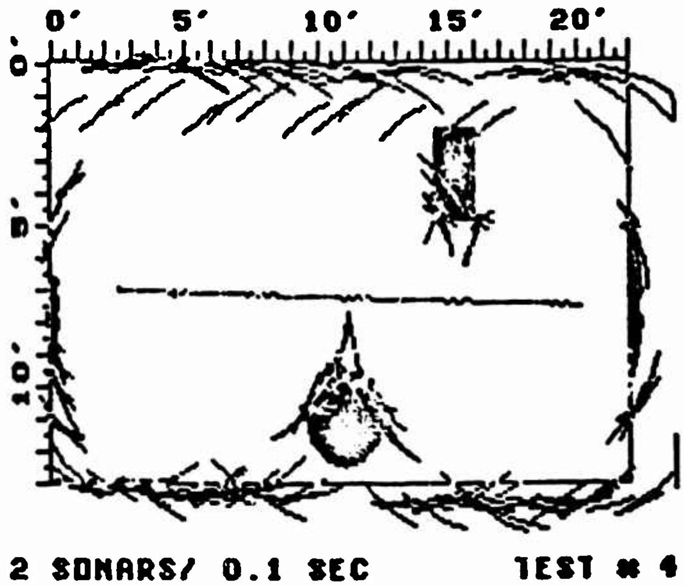


Figure 5.15 RIOT #4: Straight Two Sonars at a Time

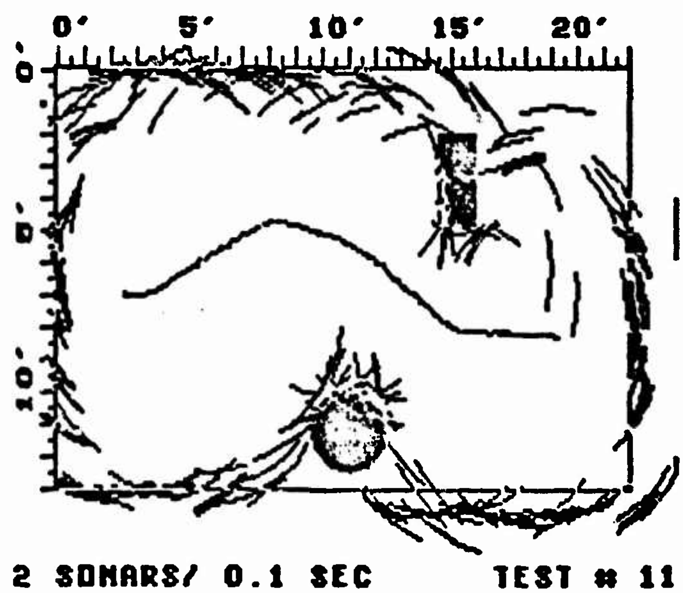


Figure 5.16 RIOT #11: Zig Zag Two Sonars at a Time

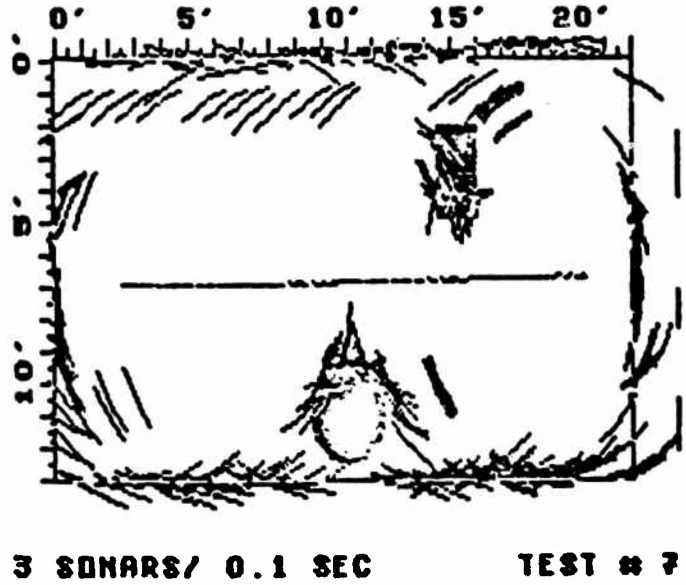


Figure 5.17 RIOT #7: Straight Three Sonars at a Time

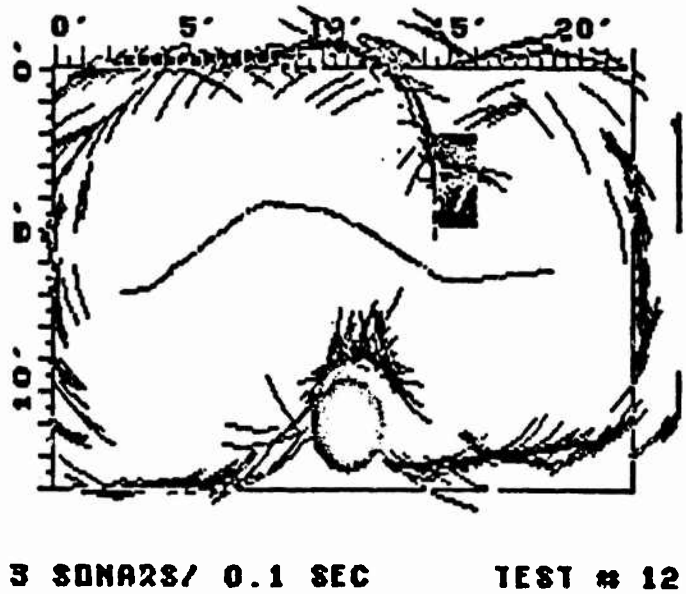


Figure 5.18 RIOT #12: Zig Zag Three Sonars at a Time

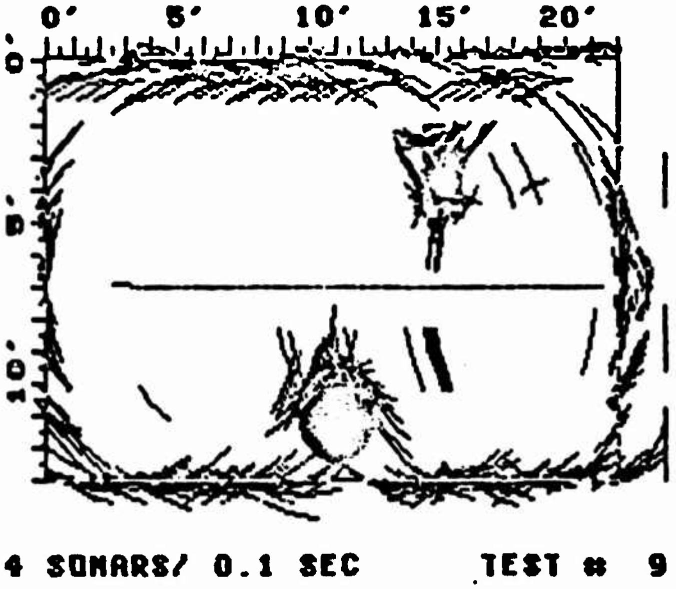


Figure 5.19 RIOT #9: Straight Four Sonars at a Time

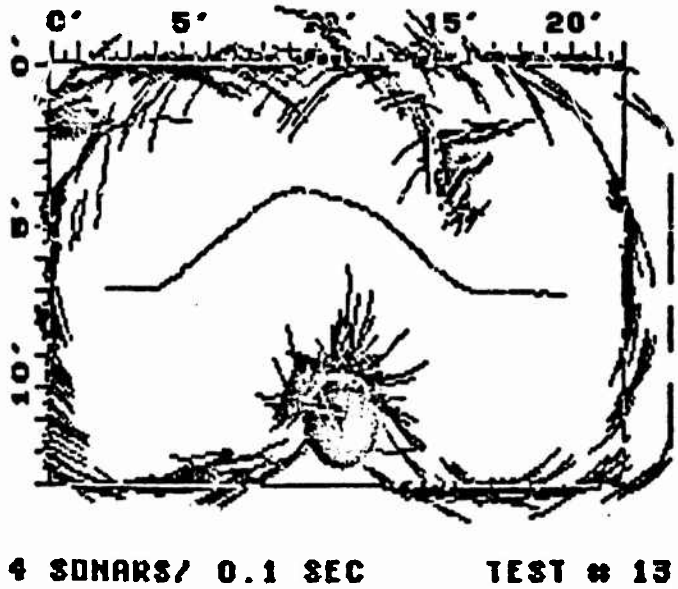


Figure 5.20 RIOT #13: Zig Zag Four Sonars at a Time

Discussion of RIOT Results

Each plot from figures 5.5 thru 5.20 represents 14 to 20 thousand eight bit words (bytes) of data like that shown in Appendix O. The data could have been encoded by the Extended Interrupt Handler (Appendix B) and reduced by a factor of four to one, however, the data reduction would have cost increased processing time by the MAPPER program of Appendix C.

Figures 5.5 thru 5.20 are accurate sonar environmental maps as compared to those of figures 5.3 and 5.4 because of the accurate heading and position information coming from the shaft encoders.

The new sonar maps show that robot navigation and obstacle avoidance in simple environments are possible from low resolution sonar vision with accurate position information.

Particular attention should be given to figure 5.9. This map identifies a skew of the sonar map with respect to the test range outline. The skew was a result of having an initial one and a half inches of right wheel travel recorded in the data (see start of Appendix O) when in fact the right wheel travel distance should have tracked the values recorded for the left wheel for straight motion (which was the case for all other straight line motion RIOTs). The difference between the right and left wheel counts was a result of having not reset the Navigation Computer after

preparing the robot for RIOT #20. The skew of figure 5.9 is therefore the result of robot operator error. It does, however, show the effect of initial heading error or isolated wheel slippage with respect to an external benchmark reference system. Note that the sonar map can be accurately created without an external reference since the map is always relative to the robot's own internal heading and position. Therefore, the map created by the robot may require a coordinate transformation to a global or external reference.

Sonar ranges greater than 7.5 feet were disreguarded for all RIOT plots since the power supply problem attenuated the maximum reliable detection for the Sonar Subsystem. Additionally, stray sonar readings are seen in a few plots. Sonar transducer C2 (figure 3.4) appears to have a maximum reliable detection range of about 7 feet and hence, the stray readings from C2 were not deleted with the 7.5 foot filter algorithm in the MAPPER program.

VI. Recommendations for Future Work on MARRS-1

Unlike some products on the market today, MARRS-1 was designed and built with modularity and expansion in mind. In addition, there are a few minor flaws in the original design that can be improved. The following are just some of the recommendations and suggestions for additional work and enhancements that will some day make MARRS-1 a truly autonomous vehicle.

1. Exercise the existing MARRS-1 robot to identify the limit of its usefulness and capabilities.

2. Transport the mapping algorithm MAPPER from an external computer to an onboard computer (either as part ofthe Navigation Computer interrupt handler or on another computer which communicates to the Navigation Computer via RS-232).

3. Design, develop, and interface an on board compass (perhaps using an A to D converter or absolute encoder) to provide MARRS-1 with instantaneous heading information. This sensor data could be used as an additional check to verify data coming from other subsystems and provide initial heading data on power-up.

4. Develop filtering algorithms that provide MARRS-1 with more accurate maps of its environment. Perhaps further characterization of the sensor subsystems is required before this can be accomplished.

5. The entire front wheel assembly requires a re-design effort. The combined mass of the yoke and the drive motor puts too much of a strain on the small gear box and shaft encoder disk mounted above. A larger frame gear box with a similar gear reduction ratio is required to eliminate false readings from the shaft encoder.

6. The MARRS-1 steering shaft feedback uses a high resolution incremental encoder to do a low resolution absolute encoder's job. A low resolution absolute encoder connected to the steering shaft by a flexible link will provide accurate information about the direction of the front wheel without excessive software/hardware overhead.

7. A better power supply for the sonar subsystem is required.

8. Integrate the electronics developed for the MARRS-1 robot structure on a larger heavier frame: a fork lift, a golf cart, or perhaps a rugged all-terrain vehicle used by the U.S. ARMY or coal mine companies.

9. Integrate Captain Glenn Monaghan's robot task planning thesis (AFIT/ENG/84D-47) with theMARRS-1.

10. Investigate the advantages/disadvantages of three wheel, four wheel, and tracked mobility configurations along with different drive and steering control systems as they apply to the mobile autonomous robot problem.

Yet Another Computer for MARRS-1

The next logical progression of MARRS-1 is to provide another computer which will control the functions of all subordinate computer systems. Its function would be much like that of a computer operating system only it would perform path planning based on sensory information fed to it from its subordinate systems and from requirements passed to it by an external computer data link. An onboard MC6809 based Radio Shack Color Computer would be the perfect addition to the droid ensemble. By adding a disk controller and disk drives, the robot becomes a stand alone hardware/software development device (minus a keyboard and monitor which may be connected remotely, or as needed).

There are several reasons for choosing the MC6809 or MC6809E over other microprocessors. First, all hardware and software developed thus far for MARRS-1 is upward compatible with the 6809. Second, with the use of OS-9 (a UNIX derivative) as the operating system, the computer can perform multi-tasking and supervise a multi-user environment. These characteristics are ideal for control of a mobile autonomous robot. Third, hardware/software availability is perhaps the most important issue as discussed in chapter II. The AFIT Mobile Robotics Lab has in its inventory (see Appendix J) a Color Computer with OS-9 and BASIC09. The Motorola Exorciser can also accommodate a 6809 card to enhance lab capabilities.

High level languages such as C open the door to increased routines and utilities that can be used to control MARRS-1. "Fortunately, the 6809 microprocessor, the OS-9 operating system, and the C language form an outstanding combination. The 6809 was specifically designed to efficiently run high level languages, and its stack-oriented instruction set and versatile repertoire of addressing modes handle the C language very well. As mentioned previously, UNIX and C are closely related, and because OS-9 is derived from UNIX, it also supports C to the degree that almost any application written in C can be transported from a UNIX system to an OS-9 system, recompiled, and correctly executed"(21:1-1). Migration toward the 68000 family is also possible by first going to the 6809 and OS-9. There is even a version of OS-9 for the 68000 family of processors making a simplified software transition guaranteed.

Cooperation Needed in Mobile Robotics

Mobile Robotics has the responsibility of bringing many technologies and disciplines together as a unified whole and has two closely related areas of research: 1) Artificial Intelligence (AI), and 2) Pattern Recognition (PR). In their report to the U.S. Army Engineer Topographic Laboratory, researchers from Stanford Research Institute (SRI) made no distinction between AI and Robotics giving instead a unified model of the two as shown in figure 6.1 (22:1).

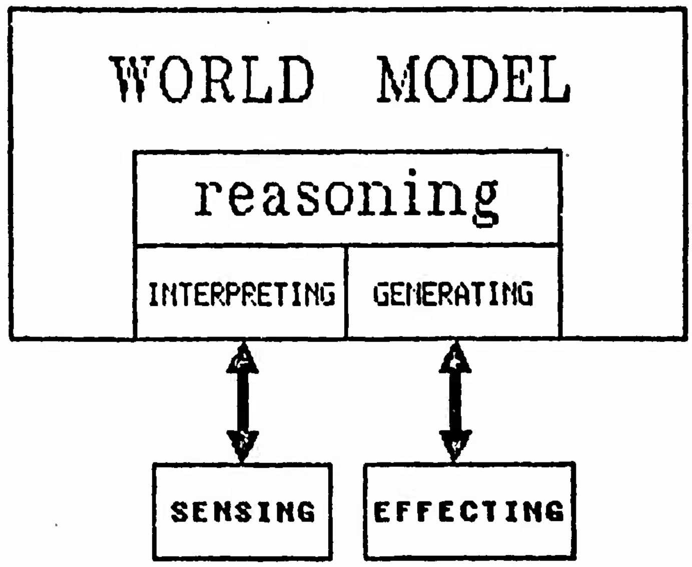


Figure 6.1 Unified AI/Robotics Model

The part of Pattern Recognition (PR) that deals with the gathering and interpreting of sensory information falls in the area of figure 6.1 labeled SENSING<--->INTERPRETING. The AFIT MARRS-1 robot in conjunction with the stereo vision system being developed by Captain James Holten of the AFIT Signal Processing lab could form the hardware of the SENSING<--->INTERPRETING and GENERATING<--->EFFECTING blocks of figure 1. Pattern Recognition would provide software for these same blocks while software for the REASONING and WORLD MODEL blocks would be provided by Artificial Intelligence. Additional sensor systems, communications, computers, software, and hardware are required to make MARRS-1 a truely autonomous robot. However, the required resources do exist within the AFIT Electrical Engineering

Department. It may be argued that it is impossible to build a truly autonomous robot since all machines created by mankind have limitations. Yet, if allowed only to use existing technology, a mobile robot with tremendous capabilities could still be realized. What is needed is a commitment from the highest level possible to solve the autonomous robot problem. The MARRS-1 robot is only a start towards that solution.

Intradepartment and interdepartment cooperation to solve problems would benefit all involved. Dr. Hans Moravec, creator of the Stanford Cart and the Carnegie-Mellon University (CMU) Rover, argues that advances in mobile robotics would bring more robust and general AI tools (23:1), (24:882). Would that not be true for all disciplines that cooperated?

Conclusion

Robots need sensors along with their mechanical devices and computational elements to be effective. This places a new perspective on the responsibilities of mobile robot designers and programmers in the way that they attack and solve their problems. These are system integration problems with demanding requirements. Overlooking an item that seems trivial may have long range negative consequences. To be sure, all that is important in mobile robotics has not been identified in this thesis.

Bibliography

1. Taylor, Captain Roslyn J. An Android Research and Development Program, MS Thesis AFIT/GE/EE/83M-3. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1983 (AD-A127 359).

2. Owen, 1Lt Randall J. III. Environmental Mapping by a Hero-1 Robot Using Sonar and a Laser Barcode Scanner, MS thesis, AFIT/GE/EE/83D-52. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1983 (AD-A138 348).

3. Craig, John. "Digital Speed Control of DC Motors," In the Beginning, New Jersey: Hayden Book Co, 1983.

4. Iijima, J, and others; "Elementary Functions of a Self-contained Robot 'Yamabico 3.1'," Proceedings of the 11th International Symposium on Industrial Robots; 7, 8, and 9 Oct, 1981.

5. Starns, Thomas W. "Design Philosophy Behind Motorola's MC68000 Part 1," Byte, 8-4: 70-91 (April 1983).

6. Starns, Thomas W. "Design Philosophy Behind Motorola's MC68000 Part 2," Byte, 8-5: 342-367 (May 1983).

7. Starns, Thomas W. "Design Philosophy Behind Motorola's MC68000 Part 3," Byte, 8-6: 339-349 (June 1983).

8. ICAM Robotics Application Guide (RAG), Air Force Wright Aeronautical Laboratories, Materials Laboratory, AFWAL TR-80-4042-VOL-2, Wright-Patterson AFB, OH, 1980 (AD-B050 811).

9. Robillard, Mark J. Microprocessor Based Robotics, Indiana. Howard W. Sams & Co., Inc., 1983.

10. Lemaire, B., and others. "A Self Adapting Low Cost Sonair for Use on Mobile Robots," Sensor Review, Oct, 1981.

11. McKean, Kevin "Computers that See," Discovery, ?:72-73, September 1984.

12. Jarvis, R. A. "A Perspective on Range Finding Techniques for Computer Vision," IEEE Transactions on Pattern Analysis and Machine Intelligence, 5:122-139 (March 1983).

13. Keller, Erik L. "Clever Robots Set to Enter Industry En Masse," Electronics, 56: 116-129 (November 17, 1983).

14. Iverson, Wesley R. "Low Cost Eyes Judge Distance," Electronics, 56: 54,56 (November 17, 1983).

15. Polaroid, Ultrasonic Ranging System, Undated.

16. Jarvis R. A. "A Laser Time-of-Flight Range Scanner for Robotic Vision," IEEE Transactions on Pattern Analysis and Machine Intelligence, 5: 505-512 (September 1983).

17. Crowley, James L. A Computational Paradigm for Three Dimensional Scene Analysis, The Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, April 1884.

18. Crowley, James L. Navigation for an Intelligent Mobile Robot, The Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, March 1984.

19. Webster's Seventh New Collegiate Dictionary, Springfield, Mass: G. & C. Merriam Co., 1972.

20. Fujiwara, N. and Tsumura, T. "An Experimental System For Processing Movement Information of Vehicle," 28th IEEE Vehicular Technology Conference Proceedings, Denver, CO, 163-168, March 1978.

21. MICROWARE C COMPILER USER'S GUIDE, Radio Shack, a Division of Tandy Corporation, 1983.

22. Brown, David R.; Fowler, Darrell V.; et al; R&D Plan for Army Applications of AI/Robotics; ETL-0296, U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia, May 1982, (AD-A118 297).

23. Elfes, Alberto; Talukdar, Sarosh N. A Distributed Control System for the CMU Rover, The Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, January 1983.

24. Moravec, Hans P. "The Stanford Cart and the CMU Rover," Proceedings of the IEEE, 71: 872-884 (July 1983).

Vita

First Lieutenant Thomas E. Clifford was born [redacted] in [redacted] where, in 1968, he graduated from [redacted] and enlistmented in the Utah Air National Guard.

He enlisted in the Regular Air Force in 1973 and became an honor graduate of the B-52 Bomb-Navigation maintenance school at Lowry AFB, Colorado. He was assigned to the 92 Bomb Wing at Fairchild AFB, Washington in 1974 and was President of the Wing Enlisted Advisory Council in 1976. In 1977 he was assigned as an IBM computer operator at the 1035th Technical Operations Group, Patrick AFB, Florida.

He received an Associate degree from the Community College of the Air Force in 1978, and a Bachelor of Science degree in Electrical Engineering from Texas Tech University in 1980 under the Airman Education and Commissioning Program. He was commissioned a Reserve Second Lieutenant in December of 1980 from Officer Training School and assigned to the 3246th Test Wing, Eglin AFB, Florida as an Electronic Warfare Test Engineer.

Lieutenant Clifford is the 1982 recipient of the Air Force Systems Command's General James Ferguson Engineering Award. He entered the Air Force Institute of Technology in May 1983 and was selected for Regular Commission status in June 1984.

Lieutenant Clifford is married to the former [redacted] of [redacted] They have five children: [redacted].

Address: [redacted]

Vita

CAPTAIN HUBERT G. SCHNEIDER III

Captain Schneider was born [redacted] Upon graduation in 1976 from [redacted] in [redacted] he matriculated as a cadet at the Virginia Military Institute, where in 1980 he completed the requirements for a B.S. in Electrical Engineering and was commissioned a Reserve Second Lieutenant of the United States Air Force.

He was then assigned as Chief, Minuteman Survival Power at the Ballistic Missile Office at Norton AFB, California where he managed the production and deployment of a new generation of high energy lithium batteries as part of the Minuteman Extended Survival Power Program.

In 1983 he entered the Masters program at the Air Force Institute of Technology, Wright Patterson AFB, Ohio and received a Regular Commission while attending the school.

He is married to [redacted] and they have one daughter, [redacted]

Address: [redacted]

```
;*********************************************
;*                                           *
;*          APPENDIX  A                      *
;*                                           *
;*********************************************
;          FILE:   NAVROM.A  MINUS NAVDEF.A
;       WHICH IS LISTED IN APPENDIX D
;
;          NOTE:  PORTIONS OF THIS LISTING DO NOT HAVE COMMENTS.
;       MOTOROLA ENGINEERING NOTE 100 IS A COMPLETE LISTING FOR MIKBUG.
;       AMERICAN MICROSYSTEMS  AMI 6800 PROTOTYPING BOARD MANUAL CONTAINS
;       A COMPLETE SOURCE CODE LISTING FOR THE PROTO PORTION OF THE CODE.
;
;
;       NAV COMPUTER ROM  AS OF MONDAY 15 OCTOBER 1984  2:30 PM
;
;**************************************************************
0000            get navhead.a
;
;                       VERSION 1.0 OF MARRSBUG
;
;            LT TOM CLIFFORD, CAPT BERT SCHNEIDER
;
;       MARRSBUG IS A ROM BASED OPERATING SYSTEM FOR THE AFIT
;
;       MOBILE AUTONOMOUS ROBOT RESEARCH SYSTEM 1 (MARRS-1).
;
;       THE ROM CONTAINS A SERIAL VERSION OF THE MOTOROLA MIKBUG
;
;       OPERATING SYSTEM AND A HIGHLY MODIFIED VERSION OF AMERICAN
;
;       MICROSYSTEMS PROTO OPERATING SYSTEM.  SYSTEM INITIALIZATION
;
;       ON POWER UP RESET OR EXTERNAL RESET CONFIGURES ALL INPUT/OUTPUT
;
;       DEVICES, PERFORMS A ROM TO RAM OVERLAY AS APPLICABLE, SIZES AND
;
;       TESTS RAM MEMORY, AND JUMPS TO THE PROTO PORTION OF THE OPERATING
;
;       SYSTEM.
;
;**************************************************************
0000            get navdef.a         ;NAV COMPUTER EQUATES AND SYS RAM USAGE
;
;       NAVDEF.A  DEFINES THE INPUT/OUTPUT AND SYSTEM RAM USAGE FOR
;       THE MARRS-1 NAVIGATION COMPUTER
;       SEE APPENDIX D FOR A COMPLETE LISTING OF NAVDEF.A
;**************************************************************
```

```
0000                    get msbug.a          ;SERIAL VERSION OF MIKBUG
                   ;
                   ;      MSBUG VERSION 1.0       22 SEPTEMBER 1984 -- LT TOM CLIFFORD
                   ;      SERIAL VERSION OF MIKBUG  (SEE MOTOROLA ENGINEERING NOTE 100)
                   ;
                   ;
                   ;
E000                      ORG 0E000H           ;FOR MIKBUG COMPATABILITY
E000   FE A0 00    MIO    LDX MIOV
E003   6E 00              JMP 00,X
E005   FE A0 06    MPOWDWN LDX MNIO
E008   6E 00              JMP 00,X
E00A   01          MLOAD  NOP
E00B   01                 NOP
E00C   01                 NOP
E00D   01                 NOP
E00E   01                 NOP
E00F   01                 NOP
E010   01                 NOP
E011   01                 NOP
E012   01                 NOP
E013   8D 63       MLOAD3 BSR MINCH
E015   81 53              CMPA #53H
E017   26 FA              BNE MLOAD3
E019   8D 5D              BSR MINCH
E01B   81 39              CMPA #39H
E01D   27 25              BEQ MC1
E01F   81 31              CMPA #31H
E021   26 F0              BNE MLOAD3
E023   7F A0 0A           CLR MCKSM
E026   8D 2D              BSR MBYTE
E028   80 02              SUBA #02
E02A   B7 A0 0B           STAA MBYTECT
E02D   8D 18              BSR MBADDR
E02F   8D 24       MLOAD11 BSR MBYTE
E031   7A A0 0B           DEC MBYTECT
E034   27 05              BEQ MLOAD15
E036   A7 00              STAA 00,X
E038   08                 INX
E039   20 F4              BRA MLOAD11
E03B   7C A0 0A    MLOAD15 INC MCKSM
E03E   27 D3              BEQ MLOAD3
E040   86 3F       MLOAD19 LDAA #3FH
E042   8D 31              BSR MOUTCH
E044   7E E0 E3    MC1 JMP MCONTRL
E047   8D 0C       MBADDR BSR MBYTE
E049   B7 A0 0C           STAA MXHI
E04C   8D 07              BSR MBYTE
E04E   B7 A0 0D           STAA MXLOW
E051   FE A0 0C           LDX MXHI
E054   39                 RTS
```

```
E055   8D 53          MBYTE   BSR MINHEX
E057   48                     ASLA
E058   48                     ASLA
E059   48                     ASLA
E05A   48                     ASLA
E05B   16                     TAB
E05C   8D 4C                  BSR MINHEX
E05E   1B                     ABA
E05F   16                     TAB
E060   FB A0 0A               ADDB MCKSM
E063   F7 A0 0A               STAB MCKSM
E066   39                     RTS
E067   44             MOUTHL  LSRA
E068   44                     LSRA
E069   44                     LSRA
E06A   44                     LSRA
E06B   84 0F          MOUTHR  ANDA #0FH
E06D   8B 30                  ADDA #30H
E06F   81 39                  CMPA #39H
E071   23 02                  BLS MOUTCH
E073   8B 07                  ADDA #07
E075   7E E1 D1       MOUTCH  JMP MOUTEEE
E078   7E E1 AC       MINCH   JMP MINEEE
E07B   8D F8          MPDATA2 BSR MOUTCH
E07D   08                     INX
E07E   A6 00          MPDATA1 LDAA 00,X
E080   81 04                  CMPA #04
E082   26 F7                  BNE MPDATA2
E084   39                     RTS
E085   8D C0          MCHANGE BSR MBADDR
E087   CE E1 9D       MCHAS1  LDX #MMCL
E08A   8D F2                  BSR MPDATA1
E08C   CE A0 0C               LDX #MXHI
E08F   8D 37                  BSR MOUT4HS
E091   FE A0 0C               LDX MXHI
E094   8D 34                  BSR MOUT2HS
E096   FF A0 0C               STX MXHI
E099   8D DD                  BSR MINCH
E09B   81 20                  CMPA #20H
E09D   26 E8                  BNE MCHAS1
E09F   8D B4                  BSR MBYTE
E0A1   09                     DEX
E0A2   A7 00                  STAA 00,X
E0A4   A1 00                  CMPA 00,X
E0A6   27 DF                  BEQ MCHAS1
E0A8   20 96                  BRA MLOAD19
E0AA   8D CC          MINHEX  BSR MINCH
E0AC   80 30                  SUBA #30H
E0AE   2B 94                  BMI MC1
E0B0   81 09                  CMPA #09
E0B2   2F 0A                  BLE MIN1HC
E0B4   81 11                  CMPA #11H
```

```
E0B6   2B 8C             BMI MC1
E0B8   81 16             CMPA #16H
E0BA   2E 88             BGT MC1
E0BC   80 07             SUBA #07
E0BE   39        MIN1HC  RTS
E0BF   A6 00     MOUT2H  LDAA 00,X
E0C1   8D [illegible]    BSR MOUTHL
E0C3   A6 00             LDAA 00,X
E0C5   08                INX
E0C6   20 A3             BRA MOUTHR
E0C8   8D F5     MOUT4HS BSR MOUT2H
E0CA   8D F3     MOUT2HS BSR MOUT2H
E0CC   86 20     MOUTS   LDAA #20H
E0CE   20 A5     MOUTS2  BRA MOUTCH
E0D0   CE BF F5          LDX #MACIAI
E0D3   8E 00 00          LDS #MACIA
E0D6   AF 00             STS 00,X
E0D8   8E E0 00          LDS #MIO
E0DB   AF 03             STS 03,X
E0DD   8E E1 13          LDS #MSFE
E0E0   AF 06             STS 06,X
E0E2   01                NOP
E0E3   8E E0 05  MCONTRL LDS #MPOWDWN
E0E6   AF 09             STS 09,X
E0E8   8E A0 42          LDS #MSTACK
E0EB   BF A0 08          STS MSP
E0EE   CE E1 9C          LDX #MMCLOFF
E0F1   8D 88             BSR MPDATA1
E0F3   8D 83             BSR MINCH
E0F5   16                TAB
E0F6   8D D4             BSR MOUTS
E0F8   C1 4C             CMPB #4CH
E0FA   26 03             BNE MCONTL2
E0FC   7E E0 0A          JMP MLOAD
E0FF   C1 4D     MCONTL2 CMPB #4DH
E101   27 82             BEQ MCHANGE
E103   C1 52             CMPB #52H
E105   27 18             BEQ MPRINT
E107   C1 50             CMPB #50H
E109   27 32             BEQ MPUNCH
E10B   C1 47             CMPB #47H
E10D   26 D4             BNE MCONTRL
E10F   BE A0 08          LDS MSP
E112   3B                RTI
E113   BF A0 08  MSFE    STS MSP
E116   30                TSX
E117   6D 06             TST 06,X
E119   26 02             BNE MSFE1
E11B   6A 05             DEC 05,X
E11D   6A 06     MSFE1   DEC 06,X
E11F   FE A0 08  MPRINT  LDX MSP
E122   08                INX
```

```
E123   8D A5                 BSR MOUT2HS
E125   8D A3                 BSR MOUT2HS
E127   8D A1                 BSR MOUT2HS
E129   8D 9D                 BSR MOUT4HS
E12B   8D 9B                 BSR MOUT4HS
E12D   CE A0 08              LDX #MSP
E130   8D 96                 BSR MOUT4HS
E132   20 AF         MC2     BRA MCONTRL
E134   0D            MMTAPE1 FCB 0DH
E135   0A                    FCB 0AH
E136   C0                    FCB 00
E137   00                    FCB 00
E138   00                    FCB 00
E139   00                    FCB 00
E13A   53                    FCB 53H
E13B   31                    FCB 31H
E13C   04                    FCB 04
E13D   86 12         MPUNCH  LDAA #12H
E13F   BD E0 75              JSR MOUTCH
E142   FE A0 02              LDX MBEGA
E145   FF A0 0F              STX MTW
E148   B6 A0 05      MPUN11  LDAA MENDA1
E14B   B0 A0 10              SUBA MTW1
E14E   F6 A0 04              LDAB MENDA
E151   F2 A0 0F              SBCB MTW
E154   26 04                 BNE MPUN22
E156   81 10                 CMPA #10H
E158   25 02                 BCS MPUN23
E15A   86 0F         MPUN22  LDAA #0FH
E15C   8B 04         MPUN23  ADDA #04
E15E   B7 A0 11              STAA MMCONT
E161   80 03                 SUBA #03
E163   B7 A0 0E              STAA MTEMP
E166   CE E1 34              LDX #MMTAPE1
E169   BD E0 7E              JSR MPDATA1
E16C   5F                    CLRB
E16D   CE A0 11              LDX #MMCONT
E170   8D 25                 BSR PUNT2
E172   CE A0 0F              LDX #MTW
E175   8D 20                 BSR PUNT2
E177   8D 1E                 BSR PUNT2
E179   FE A0 0F              LDX MTW
E17C   8D 19         MPUN32  BSR PUNT2
E17E   7A A0 0E              DEC MTEMP
E181   26 F9                 BNE MPUN32
E183   FF A0 0F              STX MTW
E186   53                    COMB
E187   37                    PSHB
E188   30                    TSX
E189   8D 0C                 BSR PUNT2
E18B   33                    PULB
E18C   FE A0 0F              LDX MTW
```

```
E18F  09              DEX
E190  BC A0 04        CPX MENDA
E193  26 B3           BNE MPUN11
E195  20 9B           BRA MC2
E197  EB 00    PUNT2  ADDB 00,X
E199  7E E0 BF        JMP MOUT2H
E19C  13       MMCLOFF FCB 13H
E19D  0D       MMCL   FCB 0DH
E19E  0A              FCB 0AH
E19F  14              FCB 14H
E1A0  00              FCB 00
E1A1  00              FCB 00
E1A2  00              FCB 00
E1A3  2A              FCB 2AH
E1A4  04              FCB 04
E1A5  FF A0 12 MSAV   STX MXTEMP
E1A8  FE BF F5        LDX MACIAI
E1AB  39              RTS
E1AC  37       MINEEE PSHB             ;INPUT WITH ECHO THRU ACIA WITH NO PARITY
E1AD  8D F6           BSR MSAV
E1AF  EE 00           LDX 00,X
E1B1  A6 00    MINOEKO LDAA 00,X
E1B3  47              ASRA
E1B4  24 FB           BCC MINOEKO
E1B6  A6 01           LDAA 01,X
E1B8  84 7F           ANDA #7FH
E1BA  36       MINOEK1 PSHA
E1BB  A6 00    MINOEK2 LDAA 00,X
E1BD  85 02           BITA #02
E1BF  27 FA           BEQ MINOEK2
E1C1  32              PULA
E1C2  A7 01           STAA 01,X
E1C4  20 24           BRA MIOS
E1C6  01              NOP
E1C7  01              NOP
E1C8  01              NOP
E1C9  01              NOP
E1CA  01              NOP
E1CB  01              NOP
E1CC  01              NOP
E1CD  01              NOP
E1CE  01              NOP
E1CF  01              NOP
E1D0  01              NOP
E1D1  37       MOUTEEE PSHB
E1D2  8D D1           BSR MSAV
E1D4  EE 00           LDX 00,X
E1D6  20 E2           BRA MINOEK1
E1D8  01              NOP
E1D9  01              NOP
E1DA  01              NOP
E1DB  01              NOP
```

```
E1DC  01                NOP
E1DD  01                NOP
E1DE  01                NOP
E1DF  01                NOP
E1E0  01                NOP
E1E1  01                NOP
E1E2  01                NOP
E1E3  01                NOP
E1E4  01                NOP
E1E5  01                NOP
E1E6  01                NOP
E1E7  01                NOP
E1E8  01                NOP
E1E9  01                NOP
E1EA  FE A0 12   MIOS   LDX MXTEMP
E1ED  33                PULB
E1EE  39                RTS
                 ;
                 ;
                 ;      END OF MSBUG  (SERIAL VERSION OF MIKBUG (SEE MOTOROLA
                 ;      ENGINEERING NOTE 100) )
                 ;
                 ;
                 ;#################################################################
E1EF                      get nbug3.a           ;HIGHLY MODIFIED VERSION OF PROTO
                 ;
                 ;                  NBUG3.A        (NEW BUG)
                 ;
                 ;      PROTO OPERATING SYSTEM VERSION 3.0
                 ;      (HIGHLY MODIFIED FOR HGS/TEC BUILT 6802/6808 COMPUTERS)
                 ;
                 ;      26 SEPTEMBER 1984 -- LT TOM CLIFFORD, CAPT BERT SCHNEIDER
                 ;      SEE AMI 6800 PROTOTYPING BOARD MANUAL FOR COMPLETE
                 ;                     SOURCE CODE LISTING
E1EF  8E BE 00   START  LDS #USRSTAK
E1F2  BF BF F1          STS SREG
E1F5  7E E1 FD          JMP START1
E1F8  7E E2 91   BREAK  JMP BREAK1
E1FB  C0 00      ACIAA  FDB IO+ACIAT
E1FD  36         START1 PSHA
E1FE  07                TPA
E1FF  B7 BF EA          STAA CREG
E202  32                PULA
E203  B7 BF EC          STAA AREG
E206  F7 BF EB          STAB BREG
E209  FF BF ED          STX XREG
E20C  BF BF F1          STS SREG
E20F  8E BE 7F          LDS #BOS
E212  CE E1 F8          LDX #BREAK
E215  FF BF FE          STX NMIVEC+1
E218  01                NOP          ;THESE THREE LINES MAY BE REPLACED WITH:
E219  01                NOP          ;        STX IRQVEC+1
```

```
E21A  01                NOP              ;THE THREE NOPS ARE USED TO KEEP A STANDARD
                                         ;CONFIGURATION OF THE ROM SUBROUTINE ADDRESSES
                                         ;FROM ONE COMPUTER TO ANOTHER RUNNING NEW BUG.
E21B  86 7E             LDAA #7EH
E21D  B7 BF F7          STAA IRQVEC
E220  B7 BF FA          STAA SWIVEC
E223  B7 BF FD          STAA NMIVEC
E226  CE E2 A8          LDX #SWI30
E229  FF BF F3          STX USWI
E22C  CE E2 95          LDX #SWIHAN
E22F  FF BF FB          STX SWIVEC+1
E232  CE E1 FB          LDX #ACIAA
E235  FF BF F5          STX ACIAI
E238  86 03             LDAA #03
E23A  B7 C0 00          STAA IO+ACIAT
E23D  86 15             LDAA #15H
E23F  B7 C0 00          STAA IO+ACIAT
E242  BD E5 2A   MONENT JSR PCRLF
E245  8E BE 7F   MONITR LDS #BOS
E248  B6 C0 01          LDAA IO+ACIAT+A.RX
E24B  86 3E             LDAA #3EH
E24D  BD E5 0B          JSR OUTCH
E250  CE BF 80          LDX #BUF
E253  FF BF D0          STX BUFPTR
E256  0D                SEC
E257  79 BF E8          ROL ECHO
E25A  8C BF C7   RT10   CPX #BUFEND
E25D  26 02             BNE RT20
E25F  20 26             BRA ABORT
E261  BD E6 11   RT20   JSR WAITTY
E264  A7 00             STAA 00,X
E266  08                INX
E267  81 0D      RT90   CMPA #0DH
E269  26 EF             BNE RT10
E26B  BD E5 AC          JSR PEXISTS
E26E  08                INX
E26F  FF BF D0          STX BUFPTR
E272  CE E8 82          LDX #CTABLE
E275  A1 00      DLOOP  CMPA 00,X
E277  26 04             BNE DL10
E279  EE 01             LDX 01,X
E27B  6E 00             JMP 00,X
E27D  08         DL10   INX
E27E  08                INX
E27F  08                INX
E280  8C E8 A3          CPX #QBADR
E283  26 F0             BNE DLOOP
E285  20 00             BRA ABORT
E287  CE E8 B9   ABORT  LDX #QQUES
E28A  8E BE 7F   MSGMON LDS #BOS
E28D  3F                SWI
E28E  12                FCB 12H
```

```
E28F   20 B1                 BRA MONENT
E291   86 80       BREAK1    LDAA #80H
E293   20 1A                 BRA SWI40
E295   30          SWIHAN    TSX
E296   EE 05                 LDX 05,X
E298   A6 00                 LDAA 00,X
E29A   2B 0C                 BMI SWI30
E29C   80 18                 SUBA #18H
E29E   2A 03                 BPL SWI20
E2A0   7E E6 5A              JMP RSRSR
E2A3   FE BF F3    SWI20     LDX USWI
E2A6   6E 00                 JMP 00,X
E2A8   30          SWI30     TSX
E2A9   6C 06                 INC 06,X
E2AB   26 02                 BNE SWI40
E2AD   6C 05                 INC 05,X
E2AF   CE BF EA    SWI40     LDX #CREG
E2B2   33          SWI50     PULB
E2B3   E7 00                 STAB 00,X
E2B5   08                    INX
E2B6   8C BF F1              CPX #SREG
E2B9   26 F7                 BNE SWI50
E2BB   AF 00                 STS 00,X
E2BD   81 81                 CMPA #81H
E2BF   26 04                 BNE PREGS
E2C1   8D 07                 BSR PR1
E2C3   20 2C                 BRA RESTAK
E2C5   8D 03       PREGS     BSR PR1
E2C7   7E E2 45              JMP MONITR
E2CA   BD E5 2A    PR1       JSR PCRLF             ;PRINT CRLF
E2CD   CE E8 EB              LDX #QREGNAM          ;POINT TO PROMPT STRING
E2D0   3F                    SWI
E2D1   12                    FCB 12H               ;AND PRINT IT
E2D2   BD E5 2A              JSR PCRLF             ;PRINT CRLF
E2D5   CE BF EA              LDX #CREG             ;POINT TO TEMPORARY REGISTERS
E2D8   C6 03                 LDAB #03              ;PRINT 3 SINGLE BYTE REGISTERS
E2DA   3F          PR10      SWI
E2DB   0F                    FCB 0FH
E2DC   BD E5 A7              JSR PSPACE            ;WITH A SPACE BETWEEN EACH PAIR
E2DF   5A                    DECB                  ;OF HEX CHARACTERS
E2E0   2E F8                 BGT PR10
E2E2   C6 03                 LDAB #03              ;NOW PRINT 3 TWO BYTE REGISTERS
E2E4   BD E5 A5    PR20      JSR P4HEXS            ;WITH A SPACE BETWEEN FOUR
E2E7   BD E5 A7              JSR PSPACE            ;HEX CHARACTERS
E2EA   5A                    DECB
E2EB   2E F7                 BGT PR20
E2ED   BD E5 2A              JSR PCRLF
E2F0   39                    RTS
E2F1   BE BF F1    RESTAK    LDS SREG
E2F4   CE BF F0              LDX #PREG+1
E2F7   A6 00       RUS10     LDAA 00,X
E2F9   36                    PSHA
```

```
E2FA  09                     DEX
E2FB  8C BF E9               CPX #TCOUNT
E2FE  26 F7                  BNE RUS10
E300  3B                     RTI
E301  B6 BF D4       CHEKSM  LDAA CKSUM
E304  36                     PSHA
E305  BD E4 CC               JSR NEX20
E308  33                     PULB
E309  53                     COMB
E30A  11                     CBA
E30B  26 01                  BNE CS1
E30D  39                     RTS
E30E  30             CS1     TSX
E30F  09                     DEX
E310  3F                     SWI
E311  0F                     FCB 0FH
E312  BD E5 A7               JSR PSPACE
E315  CE E8 BE               LDX #OCKSMER
E318  39                     RTS
E319  36             READACI PSHA                      ;LILBUG READ PORT WITHOUT WAIT
E31A  B6 C0 01               LDAA IO+ACIAT+A.RX
E31D  84 7F                  ANDA #7FH                 ;CLEAR PARITY BIT
E31F  81 17                  CMPA #17H                 ;IS IT CONTROL W ?
E321  26 02                  BNE ISCTRLX               ;IF SO WAIT FOR FURTHER INPUT
E323  3F                     SWI                       ;BEFORE CONTINUING
E324  14                     FCB 14H
E325  81 18          ISCTRLX CMPA #18H                 ;IS IT CONTROL X ?
E327  32                     PULA                      ;IF YES THEN CC IS SET
E328  39             RET39   RTS
E329  84 0F          CONV4   ANDA #0FH                 ;LILBUG CONVERT RIGHT 4 BITS AND
E32B  8B 90                  ADDA #90H                 ;PRINT ROUTINE
E32D  19                     DAA
E32E  89 40                  ADCA #40H
E330  19                     DAA
E331  3F                     SWI
E332  11                     FCB 11H
E333  39                     RTS
E334  08             ROZR2P2        INX          ;Advance compare TO pointer
E335  FF BF D9                      STX TEMP2    ;Save compare TO pointer
E338  FE BF D7       CPRMEM         LDX TEMP1    ;Fetch compare pointer
E33B  A6 00                         LDAA 0,X     ;Fetch compare data
E33D  08                            INX          ;Advance compare pointer
E33E  FF BF D7                      STX TEMP1    ;Store compare pointer
E341  FE BF D9                      LDX TEMP2    ;Fetch compare TO pointer
E344  A1 00                         CMPA 0,X     ;Compare A to indexed data
E346  26 03                         BNE CPRET    ;If not equal, return
E348  5A                            DECB         ;Decremnt byte counter
E349  26 E9                         BNE ROZR2P2  ;Counter <>0, continue comparing
E34B  39             CPRET   RTS                 ;Return with flags conditioned
E34C  BD E3 F0       DISMEM  JSR GETRNG          ;DISPLAY MEMORY ROUTINE STARTS HERE
E34F  8D 03                  BSR DISMEM1
E351  7E E2 42               JMP MONENT
```

```
E354  CE BF CC      DISMEM1 LDX #ADDL        ;POINT TO THE 'FROM' ADDRESS
E357  A6 01                 LDAA 01,X        ;GET LOW ORDER PART OF ADDRESS
E359  84 F0                 ANDA #0F0H       ;MASK LOW ORDER 4 BITS
E35B  A7 01                 STAA 01,X        ;RETURN NEW LOW ORDER PART OF ADDRESS
E35D  FE BF CE              LDX ADDH         ;GET THE 'TO' ADDRESS
E360  09                    DEX              ;MAKE IT WHAT THE USER SAID IT WAS TO BE
E361  3F                    SWI              ;
E362  02                    FCB 02H          ;
E363  CA 0F                 ORAB #0FH        ;SET LOW ORDER 4 BITS
E365  3F                    SWI              ;
E366  03                    FCB 03H          ;
E367  FF BF CE              STX ADDH         ;RESTORE NEW 'TO' ADDRESS
E36A  BD E5 2A      BYTENUM JSR PCRLF        ;PRINT BYTE NUMBER GUIDE LINE BY FIRST PRINTING CRLF
E36D  BD E5 2A              JSR PCRLF        ;TWICE
E370  CE E8 E4              LDX #06SPACE     ;PRINT 6 LEADING SPACES
E373  3F                    SWI
E374  12                    FCB 12H
E375  4F                    CLRA
E376  36            BYTENM1 PSHA             ;CONVERT RIGHT 4 BITS AND PRINT AS ASCII
E377  BD E3 29              JSR CONV4        ;UPPER CASE CHARACTER
E37A  BD E5 A7              JSR PSPACE       ;PRINT TWO
E37D  BD E5 A7              JSR PSPACE       ;SPACES
E380  32                    PULA             ;GET COUNTER
E381  4C                    INCA             ;HAVE WE PRINTED
E382  81 10                 CMPA #10H        ;HEX NUMBERS 0 THRU F?
E384  26 F0                 BNE BYTENM1      ;KEEP PRINTING THEM IF NOT FINISHED
E386  8D 91         OUTLOOP BSR READACI      ;CHECK FOR PRINT END OR WAIT FROM USER
E388  27 57                 BEQ ENDDM        ;END PRINT IF CONTROL X KEY WAS HIT
E38A  BD E5 2A              JSR PCRLF        ;ELSE CONTINUE PRINT WITH CRLF
E38D  CE BF CC              LDX #ADDL        ;GET ADDRESS OF DATA TO BE PRINTED
E390  BD E5 A5              JSR P4HEXS       ;PRINT ADDRESS AS 4 HEX WITH A SPACE
E393  FE BF CC              LDX ADDL         ;POINT TO THE DATA
E396  C6 10                 LDAB #10H        ;SET UP A COUNTER FOR 16 BYTES
E398  3F            INLOOP1 SWI              ;PRINT BYTE AS TWO HEX AND POINT TO NEXT BYTE
E399  0F                    FCB 0FH
E39A  BD E5 A7              JSR PSPACE       ;PRINT A SPACE
E39D  5A                    DECB             ;ONE LESS BYTE TO PRINT
E39E  26 F8                 BNE INLOOP1      ;PRINT NEXT BYTE IF NOT DONE
E3A0  BD E5 A7              JSR PSPACE       ;PRINT ANOTHER SPACE
E3A3  C6 10                 LDAB #10H        ;NOW PRINT THE 16 BYTES AS ASCII IF POSSIBLE
E3A5  FE BF CC              LDX ADDL         ;POINT TO THE FIRST BYTE
E3A8  A6 00         INLOOP2 LDAA 00,X        ;GET THE BYTE
E3AA  84 7F                 ANDA #7FH        ;STRIP OFF EIGHTH BIT
E3AC  81 20                 CMPA #20H        ;IF BYTE IS LESS THAN ASCII SPACE
E3AE  2D 04                 BLT PERIOD       ;PRINT A PERIOD
E3B0  81 61                 CMPA #61H        ;ELSE CHECK TO SEE IF IT IS A PRINTABLE CHARACTER
E3B2  2D 02                 BLT PRNTASC
E3B4  86 2E         PERIOD  LDAA #2EH        ;NOT PRINTABLE SO PRINT A PERIOD
E3B6  3F            PRNTASC SWI              ;PRINT THE CHARACTER OR A PERIOD
E3B7  11                    FCB 11H
E3B8  08                    INX              ;POINT TO NEXT BYTE
E3B9  5A                    DECB             ;ONE LESS TO PRINT
```

```
E3BA  26 EC            BNE INLOOP2      ;GO PRINT THE REST OF THE 16 BYTES
E3BC  09               DEX              ;POINT TO THE LAST BYTE PRINTED
E3BD  FF BF CC         STX ADDL         ;SAVE THE ADDRESS
E3C0  CE BF CC         LDX #ADDL        ;GET A POINTER TO THE LAST BYTE PRINTED
E3C3  FF BF D7         STX TEMP1        ;SAVE THE POINTER FOR COMPARISON
E3C6  CE BF CE         LDX #ADDH        ;GET A POINTER TO THE LAST BYTE TO BE PRINTED
E3C9  FF BF D9         STX TEMP2        ;SAVE THE POINTER FOR COMPARISON
E3CC  C6 02            LDAB #02         ;THE POINTERS ARE 2 BYTES EACH
E3CE  BD E3 38         JSR CPRMEM       ;COMPARE ADDRESSES
E3D1  22 0E            BHI ENDDM        ;BRANCH IF FINISHED WITH MEMORY PRINT
E3D3  FE BF CC         LDX ADDL         ;ELSE GET THE ADDRESS OF LAST BYTE PRINTED
E3D6  08               INX              ;AND POINT TO NEXT BYTE TO BE PRINTED
E3D7  FF BF CC         STX ADDL         ;SAVE THE ADDRESS
E3DA  7D BF CD         TST ADDL+1       ;IF WE HAVE PRINTED A BLOCK OF DATA
E3DD  26 A7            BNE OUTLOOP      ;THEN GO PRINT THE BYTE NUMBER GUIDE LINE AND 16 BYTES
E3DF  20 89            BRA BYTENUM      ;ELSE GO PRINT NEXT 16 BYTES
E3E1  39        ENDDM  RTS              ;RETURN TO CALLER
E3E2  8D 03     EOF    BSR EOF1
E3E4  7E E2 42         JMP MONENT
E3E7  CE E8 D0  EOF1   LDX #QTPEOF
E3EA  3F               SWI
E3EB  12               FCB 12H
E3EC  BD E5 2A         JSR PCRLF
E3EF  39               RTS
E3F0  BD E4 E6  GETRNG JSR NXTADR
E3F3  FE BF CA         LDX ADR
E3F6  FF BF CC         STX ADDL
E3F9  FF BF CE         STX ADDH
E3FC  BD E4 E6         JSR NXTADR
E3FF  27 06            BEQ GETRG3
E401  FE BF CA  GETRG1 LDX ADR
E404  FF BF CE         STX ADDH
E407  CE BF 80  GETRG3 LDX #BUF
E40A  A6 4E            LDAA 4EH,X
E40C  E6 4F            LDAB 4FH,X
E40E  E0 4D            SUBB 4DH,X
E410  A2 4C            SBCA 4CH,X
E412  24 04            BCC GETRG4
E414  CE E8 AB  RNGERR LDX #QRNGERR
E417  39               RTS
E418  FE BF CE  GETRG4 LDX ADDH
E41B  08               INX
E41C  FF BF CE         STX ADDH
E41F  39               RTS
E420  BD E4 E6  GO     JSR NXTADR
E423  27 06            BEQ GO10
E425  FE BF CA         LDX ADR
E428  FF BF EF         STX PREG
E42B  7E E2 F1  GO10   JMP RESTAK
E42E  8D 03     LOAD   BSR LOAD2
E430  7E E2 8A         JMP MSGMON
E433  CE 00 00  LOAD2  LDX #XZERO
```

```
E436   FF BF C8            STX OFFSET
E439   FF BF CC            STX ADDL
E43C   09         LOOFST   DEX
E43D   FF BF CE            STX ADDH
E440   BD E4 E6            JSR NXTADR
E443   27 1E               BEQ LOAD1
E445   FE BF CA            LDX ADR
E448   FF BF C8            STX OFFSET
E44B   BD E4 E6            JSR NXTADR
E44E   27 13               BEQ LOAD1
E450   FE BF C8            LDX OFFSET
E453   FF BF CC            STX ADDL
E456   CE 00 00            LDX #XZERO
E459   FF BF C8            STX OFFSET
E45C   8D A3               BSR GETR01
E45E   FE BF CE            LDX ADDH
E461   20 D9               BRA LOOFST
E463   01         LOAD1    NOP
E464   8D 5D      RDPRE    BSR FINDS
E466   BD E6 11            JSR WAITTY
E469   81 30               CMPA #30H
E46B   27 F7               BEQ RDPRE
E46D   B7 BF D2            STAA RECTYP
E470   7F BF D4            CLR CKSUM
E473   BD E4 CC            JSR NEX2D
E476   4A                  DECA
E477   4A                  DECA
E478   4A                  DECA
E479   B7 BF D3            STAA COUNT
E47C   BD E4 CC            JSR NEX2D
E47F   B7 BF CA            STAA ADR
E482   BD E4 CC            JSR NEX2D
E485   BB BF C9            ADDA OFFSET+1
E488   B7 BF CB            STAA ADR+1
E48B   B6 BF CA            LDAA ADR
E48E   B9 BF C8            ADCA OFFSET
E491   B7 BF CA            STAA ADR
E494   B6 BF D2            LDAA RECTYP
E497   81 31               CMPA #31H
E499   26 14               BNE LHF4
E49B   BD E4 CC   LDR10    JSR NEX2D
E49E   FE BF CA            LDX ADR
E4A1   BD E5 C2            JSR SETOFF
E4A4   08                  INX
E4A5   FF BF CA            STX ADR
E4A8   7A BF D3            DEC COUNT
E4AB   2E EE               BGT LDR10
E4AD   20 04               BRA LHF9
E4AF   81 39      LHF4     CMPA #39H
E4B1   26 0E               BNE BADTAP
E4B3   BD E3 01   LHF9     JSR CHEKSM
E4B6   B6 BF D2            LDAA RECTYP
```

```
E4B9  81 39             CMPA #39H
E4BB  26 A6             BNE LOAD1
E4BD  CE E8 B5          LDX #QEOF
E4C0  39                RTS
E4C1  20 A1     BADTAP  BRA RDPRE
E4C3  01        FINDS   NOP
E4C4  BD E6 11  FS10    JSR WAITTY
E4C7  81 53             CMPA #53H
E4C9  26 F9             BNE FS10
E4CB  39                RTS
E4CC  BD E6 11  NEX20   JSR WAITTY
E4CF  16                TAB
E4D0  BD E6 11          JSR WAITTY
E4D3  36                PSHA
E4D4  37                PSHB
E4D5  30                TSX
E4D6  C6 02             LDAB #02
E4D8  3F                SWI
E4D9  15                FCB 15H
E4DA  24 E5             BCC BADTAP
E4DC  17                TBA
E4DD  FB BF D4          ADDB CKSUM
E4E0  F7 BF D4          STAB CKSUM
E4E3  31                INS
E4E4  31                INS
E4E5  39                RTS
E4E6  7F BF CA  NXTADR  CLR ADR
E4E9  7F BF CB          CLR ADR+1
E4EC  BD E5 AC          JSR PEXISTS
E4EF  26 01             BNE NA1
E4F1  39        NXTRTS  RTS
E4F2  C6 47     NA1     LDAB #47H
E4F4  3F                SWI
E4F5  15                FCB 15H
E4F6  FF BF D0          STX BUFPTR
E4F9  B7 BF CA          STAA ADR
E4FC  F7 BF CB          STAB ADR+1
E4FF  A6 00             LDAA 00,X
E501  3F                SWI
E502  13                FCB 13H
E503  25 01             BCS NA3
E505  39                RTS
E506  7E E2 87  NA3     JMP ABORT
E509  A6 00     OUTCHX  LDAA 00,X
E50B  37        OUTCH   PSHB
E50C  F6 C0 00          LDAB IO+ACIAT
E50F  57                ASRB
E510  24 0A             BCC OC10
E512  F6 C0 01          LDAB IO+ACIAT+A.RX
E515  C1 1B             CMPB #1BH
E517  26 03             BNE OC10
E519  7E E2 87          JMP ABORT
```

```
E51C  3F          OC10    SWI
E51D  11                  FCB 11H
E51E  81 0D               CMPA #0DH
E520  26 06               BNE OC20
E522  86 0A               LDAA #0AH
E524  3F                  SWI
E525  11                  FCB 11H
E526  86 0D               LDAA #0DH
E528  33          OC20    PULB
E529  39                  RTS
E52A  86 0D       PCRLF   LDAA #0DH
E52C  20 DD               BRA OUTCH
E52E  8D 03       PUNCH   BSR PUNCH1
E530  7E E2 42            JMP MONENT
E533  BD E3 F0    PUNCH1  JSR GETRNG
E536  CE 00 06            LDX #XZERO
E539  FF BF C8            STX OFFSET
E53C  8D A8               BSR NXTADR
E53E  27 06               BEQ PHF20
E540  FE BF CA            LDX ADR
E543  FF BF C8            STX OFFSET
E546  F6 BF CF    PHF20   LDAB ADDH+1
E549  F0 BF CD            SUBB ADDL+1
E54C  B6 BF CE            LDAA ADDH
E54F  B2 BF CC            SBCA ADDL
E552  26 04               BNE PUND10
E554  C1 10               CMPB #10H
E556  23 02               BLS PUND20
E558  C6 10       PUND10  LDAB #10H
E55A  5C          PUND20  INCB
E55B  5C                  INCB
E55C  5C                  INCB
E55D  F7 BF D3            STAB COUNT
E560  CE E8 DB            LDX #QSI
E563  3F                  SWI
E564  12                  FCB 12H
E565  5F                  CLRB
E566  CE BF D3            LDX #COUNT
E569  8D 32               BSR PUNBYTE
E56B  37                  PSHB
E56C  FE BF CC            LDX ADDL
E56F  B6 BF C8            LDAA OFFSET
E572  F6 BF C9            LDAB OFFSET+1
E575  3F                  SWI
E576  08                  FCB 08
E577  FF BF CA            STX ADR
E57A  CE BF CA            LDX #ADR
E57D  33                  PULB
E57E  8D 1D               BSR PUNBYTE
E580  8D 1B               BSR PUNBYTE
E582  FE BF CC            LDX ADDL
E585  8D 16       PREC10  BSR PUNBYTE
```

```
E587   2E FC              BGT PREC10
E589   FF BF CC           STX ADDL
E58C   CE BF D4           LDX #CKSUM
E58F   53                 COMB
E590   E7 00              STAB 00,X
E592   8D 09              BSR PUNBYTE
E594   FE BF CC           LDX ADDL
E597   BC BF CE           CPX ADDH
E59A   26 AA              BNE PHF20
E59C   39                 RTS
E59D   EB 00      PUNBYTE ADDB 00,X
E59F   3F                 SWI
E5A0   0F                 FCB 0FH
E5A1   7A BF D3           DEC COUNT
E5A4   39                 RTS
E5A5   3F         P4HEXS  SWI
E5A6   10                 FCB 10H
E5A7   86 20      PSPACE  LDAA #20H
E5A9   3F                 SWI
E5AA   11                 FCB 11H
E5AB   39                 RTS
E5AC   FE BF D0   PEXISTS LDX BUFPTR
E5AF   A6 00      PXISTX  LDAA 00,X
E5B1   3F                 SWI
E5B2   13                 FCB 13H
E5B3   25 07              BCS PX2
E5B5   81 0D              CMPA #0DH
E5B7   27 03              BEQ PX2
E5B9   08                 INX
E5BA   20 F3              BRA PXISTX
E5BC   FF BF D0   PX2     STX BUFPTR
E5BF   81 0D              CMPA #0DH
E5C1   39                 RTS
E5C2   36         SETOFF  PSHA
E5C3   B6 BF CC           LDAA ADDL
E5C6   F6 BF CD           LDAB ADDL+1
E5C9   3F                 SWI
E5CA   0B                 FCB 0BH
E5CB   22 0A              BHI SETOUT
E5CD   B6 BF CE           LDAA ADDH
E5D0   F6 BF CF           LDAB ADDH+1
E5D3   3F                 SWI
E5D4   0B                 FCB 0BH
E5D5   24 07              BCC SETPUL
E5D7   32         SETOUT  PULA
E5D8   86 FF              LDAA #0FFH
E5DA   3F                 SWI
E5DB   11                 FCB 11H
E5DC   20 15              BRA SETM1
E5DE   32         SETPUL  PULA
E5DF   A7 00      SETMEM  STAA 00,X
E5E1   A1 00              CMPA 00,X
```

```
E5E3  27 0E           BEQ SETM1
E5E5  FF BF CA        STX ADR
E5E8  CE BF CA        LDX #ADR
E5EB  8D B8           BSR P4HEXS
E5ED  CE E8 A3  PBADR LDX #QBADR
E5F0  7E E2 8A        JMP MSGMON
E5F3  39        SETM1 RTS
E5F4  BD E4 E6  SM    JSR NXTADR
E5F7  FE BF CA        LDX ADR
E5FA  FF BF CC        STX ADDL
E5FD  BD E4 E6  SM10  JSR NXTADR
E600  27 0C           BEQ SM30
E602  FE BF CC        LDX ADDL
E605  17              TBA
E606  8D D7           BSR SETMEM
E608  08              INX
E609  FF BF CC        STX ADDL
E60C  20 EF           BRA SM10
E60E  7E E2 45  SM30  JMP MONITR
E611  3F        WAITTY SWI
E612  14              FCB 14H
E613  81 1B           CMPA #1BH
E615  26 03           BNE W20
E617  7E E2 87        JMP ABORT
E61A  81 7F     W20   CMPA #7FH
E61C  27 F3           BEQ WAITTY
E61E  7D BF E8        TST ECHO
E621  27 03           BEQ W30
E623  BD E5 0B        JSR OUTCH
E626  39        W30   RTS
E627  FE BF CC  INK   LDX ADDL
E62A  08              INX
E62B  FF BF CC        STX ADDL
E62E  BC BF CE        CPX ADDH
E631  26 02           BNE ADDRS
E633  31              INS
E634  31              INS
E635  39        ADDRS RTS
E636  7E E5 ED  JBAD  JMP PBADR
E639  8D 03     MOVE  BSR MOVE2
E63B  7E E2 45        JMP MONITR
E63E  BD E3 F0  MOVE2 JSR GETRNG
E641  BD E4 E6        JSR NXTADR
E644  27 F0           BEQ JBAD
E646  FE BF CC  MOVE1 LDX ADDL
E649  A6 00           LDAA 00,X
E64B  FE BF CA        LDX ADR
E64E  BD E5 DF        JSR SETMEM
E651  08              INX
E652  FF BF CA        STX ADR
E655  BD E6 27        JSR INK
E658  20 EC           BRA MOVE1
```

```
E65A   30          RSRSR   TSX
E65B   EE 05               LDX 05,X
E65D   4F                  CLRA
E65E   E6 00               LDAB 00,X
E660   58                  ASLB
E661   49                  ROLA
E662   8D 00               BSR LOCW
E664   30          LOCW    TSX
E665   EB 01               ADDB 01,X
E667   A9 00               ADCA 00,X
E669   CB 24               ADDB #24H
E66B   89 01               ADCA #01
E66D   A7 00               STAA 00,X
E66F   E7 01               STAB 01,X
E671   EE 00               LDX 00,X
E673   EB 01               ADDB 01,X
E675   A9 00               ADCA 00,X
E677   30                  TSX
E678   A7 00               STAA 00,X
E67A   E7 01               STAB 01,X
E67C   A6 02               LDAA 02,X
E67E   06                  TAP
E67F   EE 00               LDX 00,X
E681   31                  INS
E682   31                  INS
E683   AD 00               JSR 00,X
E685   30                  TSX
E686   6C 06               INC 06,X
E688   26 02               BNE RSRXIT
E68A   6C 05               INC 05,X
E68C   3B          RSRXIT  RTI
E68D   34                  DES
E68E   34                  DES
E68F   34                  DES
E690   34                  DES
E691   34                  DES
E692   C6 09               LDAB #09
E694   30                  TSX
E695   A6 05       PS      LDAA 05,X
E697   A7 00               STAA 00,X
E699   08                  INX
E69A   5A                  DECB
E69B   26 F8               BNE PS
E69D   C6 05               LDAB #05
E69F   30                  TSX
E6A0   A6 02       PC      LDAA 02,X
E6A2   A7 09               STAA 09,X
E6A4   08                  INX
E6A5   5A                  DECB
E6A6   26 F8               BNE PC
E6A8   39                  RTS
E6A9   30                  TSX
```

```
E6AA   C6 05          LDAB #05
E6AC   A6 09   TRC    LDAA 09,X
E6AE   A7 02          STAA 02,X
E6B0   08             INX
E6B1   5A             DECB
E6B2   26 F8          BNE TRC
E6B4   C6 09          LDAB #09
E6B6   A6 03   RS     LDAA 03,X
E6B8   A7 08          STAA 08,X
E6BA   09             DEX
E6BB   5A             DECB
E6BC   26 F8          BNE RS
E6BE   31             INS
E6BF   31             INS
E6C0   31             INS
E6C1   31             INS
E6C2   31             INS
E6C3   39             RTS
E6C4   30             TSX
E6C5   A6 05          LDAA 05,X
E6C7   E6 06          LDAB 06,X
E6C9   A7 04   STAB   STAA 04,X
E6CB   E7 03          STAB 03,X
E6CD   39             RTS
E6CE   30             TSX
E6CF   A6 04   TABX1  LDAA 04,X
E6D1   A7 05          STAA 05,X
E6D3   A6 03          LDAA 03,X
E6D5   A7 06          STAA 06,X
E6D7   39             RTS
E6D8   30             TSX
E6D9   A6 05   XABX1  LDAA 05,X
E6DB   36             PSHA
E6DC   E6 06          LDAB 06,X
E6DE   8D EF          BSR TABX1
E6E0   32             PULA
E6E1   20 E6          BRA STAB
E6E3   34             DES
E6E4   34             DES
E6E5   30             TSX
E6E6   86 09          LDAA #09
E6E8   E6 02   MA     LDAB 02,X
E6EA   E7 00          STAB 00,X
E6EC   08             INX
E6ED   4A             DECA
E6EE   26 F8          BNE MA
E6F0   30             TSX
E6F1   A6 05          LDAA 05,X
E6F3   A7 09          STAA 09,X
E6F5   A6 06          LDAA 06,X
E6F7   A7 0A          STAA 0AH,X
E6F9   39             RTS
```

```
E6FA   30             TSX
E6FB   A6 09          LDAA 09,X
E6FD   A7 05          STAA 05,X
E6FF   A6 0A          LDAA 0AH,X
E701   A7 06          STAA 06,X
E703   86 09          LDAA #09
E705   E6 08   PULXA  LDAB 08,X
E707   E7 0A          STAB 0AH,X
E709   09             DEX
E70A   4A             DECA
E70B   26 F8          BNE PULXA
E70D   31             INS
E70E   31             INS
E70F   39             RTS
E710   30             TSX
E711   8D C6          BSR XABX1
E713   8D 03          BSR ADDABX1
E715   20 C2          BRA XABX1
E717   30             TSX
E718   A6 03   ADDABX1 LDAA 03,X
E71A   E6 04          LDAB 04,X
E71C   AB 06   ADDAB  ADDA 06,X
E71E   A7 06          STAA 06,X
E720   E9 05          ADCB 05,X
E722   07      STAUXH TPA
E723   E7 05          STAB 05,X
E725   6D 06          TST 06,X
E727   27 02   TESTZ  BEQ TA
E729   84 FB          ANDA #0FBH
E72B   A7 02   TA     STAA 02,X
E72D   39             RTS
E72E   30             TSX
E72F   A6 04          LDAA 04,X
E731   C6 00   ADDZ   LDAB #00
E733   20 E7          BRA ADDAB
E735   30             TSX
E736   A6 03          LDAA 03,X
E738   20 F7          BRA ADDZ
E73A   30             TSX
E73B   8D 9C          BSR XABX1
E73D   8D 03          BSR SUBABX1
E73F   20 98          BRA XABX1
E741   30             TSX
E742   E6 05   SUBABX1 LDAB 05,X
E744   A6 06          LDAA 06,X
E746   A0 03          SUBA 03,X
E748   A7 06          STAA 06,X
E74A   E2 04          SBCB 04,X
E74C   20 D4          BRA STAUXH
E74E   30             TSX
E74F   E6 04          LDAB 04,X
E751   A6 06   TSUB   LDAA 06,X
```

```
E753  10          SBA
E754  A7 06       STAA 06,X
E756  E6 05       LDAB 05,X
E758  C2 00       SBCB #00
E75A  20 C6       BRA STAUXH
E75C  30          TSX
E75D  E6 03       LDAB 03,X
E75F  20 F0       BRA TSUB
E761  8D 11       BSR MPY8
E763  37          PSHB
E764  16          TAB
E765  32          PULA
E766  30          TSX
E767  20 B3       BRA ADDAB
E769  8D 09       BSR MPY8
E76B  30          TSX
E76C  E7 03       STAB 03,X
E76E  A7 04       STAA 04,X
E770  07          TPA
E771  5D          TSTB
E772  20 B3       BRA TESTZ
E774  86 08  MPY8    LDAA #08
E776  36          PSHA
E777  4F          CLRA
E778  30          TSX
E779  E6 06       LDAB 06,X
E77B  56          RORB
E77C  24 02  MPYLP   BCC MPYSHFT
E77E  AB 07       ADDA 07,X
E780  46     MPYSHFT RORA
E781  56          RORB
E782  6A 00       DEC 00,X
E784  26 F6       BNE MPYLP
E786  31          INS
E787  39          RTS
E788  FF          FCB 0FFH
E789  05          FCB 05
E78A  FF          FCB 0FFH
E78B  1F          FCB 1FH
E78C  FF          FCB 0FFH
E78D  38          FCB 38H
E78E  FF          FCB 0FFH
E78F  40          FCB 40H
E790  FF          FCB 0FFH
E791  48          FCB 48H
E792  FF          FCB 0FFH
E793  51          FCB 51H
E794  FF          FCB 0FFH
E795  66          FCB 66H
E796  FF          FCB 0FFH
E797  7A          FCB 7AH
E798  FF          FCB 0FFH
```

```
E799   7F                 FCB 7FH
E79A   FF                 FCB 0FFH
E79B   94                 FCB 94H
E79C   FF                 FCB 0FFH
E79D   99                 FCB 99H
E79E   FF                 FCB 0FFH
E79F   9C                 FCB 9CH
E7A0   FF                 FCB 0FFH
E7A1   A1                 FCB 0A1H
E7A2   FF                 FCB 0FFH
E7A3   AC                 FCB 0ACH
E7A4   FF                 FCB 0FFH
E7A5   B8                 FCB 0B8H
E7A6   00                 FCB 00
E7A7   17                 FCB 17H
E7A8   00                 FCB 00
E7A9   10                 FCB 10H
E7AA   00                 FCB 00
E7AB   2F                 FCB 2FH
E7AC   00                 FCB 00
E7AD   4F                 FCB 4FH
E7AE   00                 FCB 00
E7AF   5D                 FCB 5DH
E7B0   00                 FCB 00
E7B1   86                 FCB 86H
E7B2   00                 FCB 00
E7B3   96                 FCB 96H
E7B4   FF                 FCB 0FFH
E7B5   AD                 FCB 0ADH
E7B6   FF                 FCB 0FFH
E7B7   B3                 FCB 0B3H
E7B8   30                 TSX
E7B9   EE 05              LDX 05,X
E7BB   8D 06              BSR PHEX
E7BD   30                 TSX
E7BE   EE 05              LDX 05,X
E7C0   8D 01              BSR PHEX
E7C2   39                 RTS
E7C3   A6 00     PHEX     LDAA 00,X
E7C5   8D 29              BSR ASCIIR
E7C7   36                 PSHA
E7C8   A6 00              LDAA 00,X
E7CA   8D 20              BSR ASCIIL
E7CC   8D 0E              BSR PUTAX
E7CE   32                 PULA
E7CF   8D 10              BSR PUTA
E7D1   30        PINCX    TSX
E7D2   6C 08              INC 08,X
E7D4   26 02              BNE PINXRTS
E7D6   6C 07              INC 07,X
E7D8   39        PINXRTS  RTS
E7D9   30                 TSX
```

```
E7DA   A6 04              LDAA 04,X
E7DC   FE BF F5    PUTAX  LDX ACIAI
E7DF   EE 00              LDX 00,X
E7E1   36          PUTA   PSHA
E7E2   A6 00       PUTRDY LDAA 00,X
E7E4   85 02              BITA #02
E7E6   27 FA              BEQ PUTRDY
E7E8   32                 PULA
E7E9   A7 01              STAA 01,X
E7EB   39                 RTS
E7EC   44          ASCIIL LSRA
E7ED   44                 LSRA
E7EE   44                 LSRA
E7EF   44                 LSRA
E7F0   84 0F       ASCIIR ANDA #0FH
E7F2   8B 30              ADDA #30H
E7F4   81 39              CMPA #39H
E7F6   23 02              BLS ASCRTS
E7F8   8B 07              ADDA #07
E7FA   39          ASCRTS RTS
E7FB   30          PMESS  TSX
E7FC   EE 05              LDX 05,X
E7FE   A6 00              LDAA 00,X
E800   81 04              CMPA #04
E802   27 06              BEQ PMSRTS
E804   8D D6              BSR PUTAX
E806   8D C9              BSR PINCX
E808   20 F1              BRA PMESS
E80A   39          PMSRTS RTS
E80B   30                 TSX
E80C   EE 05              LDX 05,X
E80E   8D 05              BSR ALPNUM
E810   07          SCARRY TPA
E811   30          SETUS  TSX
E812   A7 02              STAA 02,X
E814   39                 RTS
E815   A6 00       ALPNUM LDAA 00,X
E817   81 41              CMPA #41H
E819   2D 0E              BLT ANNUM
E81B   81 5A              CMPA #5AH
E81D   2E 12              BGT ANNOTOK
E81F   81 C7              CMPA #0C7H
E821   29 10              BVS ANRTS
E823   80 07              SUBA #07
E825   84 0F       ANOK   ANDA #0FH
E827   0D                 SEC
E828   39                 RTS
E829   81 30       ANNUM  CMPA #30H
E82B   2D 04              BLT ANNOTOK
E82D   81 39              CMPA #39H
E82F   2F F4              BLE ANOK
E831   0C          ANNOTOK CLC
```

```
E832  0B                SEV
E833  39        ANRTS   RTS
E834  20 9B     JPINCX  BRA PINCX
E836  FE BF F5          LDX ACIAI
E839  EE 00             LDX 00,X
E83B  A6 00     INWAIT  LDAA 00,X
E83D  47                ASRA
E83E  24 FB             BCC INWAIT
E840  A6 01             LDAA 01,X
E842  84 7F             ANDA 07FH
E844  30                TSX
E845  A7 04             STAA 04,X
E847  39                RTS
E848  30                TSX
E849  E6 03             LDAB 03,X
E84B  6F 04             CLR 04,X
E84D  6F 03             CLR 03,X
E84F  30        CONHB   TSX
E850  EE 05             LDX 05,X
E852  8D C1             BSR ALPNUM
E854  25 09             BCS CONFND
E856  5A                DECB
E857  2F 04             BLE CONENDC
E859  8D D9             BSR JPINCX
E85B  20 F2             BRA CONHB
E85D  20 B1     CONENDC BRA SCARRY
E85F  30        CONFND  TSX
E860  EE 05             LDX 05,X
E862  8D B1             BSR ALPNUM
E864  29 18             BVS CONNOGD
E866  37                PSHB
E867  C6 04             LDAB #04
E869  30                TSX
E86A  68 04     CONSLP  ASL 04,X
E86C  69 05             ROL 05,X
E86E  5A                DECB
E86F  2E F9             BGT CONSLP
E871  AA 04             ORAA 04,X
E873  A7 04             STAA 04,X
E875  33                PULB
E876  8D BC             BSR JPINCX
E878  5A                DECB
E879  2E E4             BGT CONFND
E87B  0D                SEC
E87C  20 92             BRA SCARRY
E87E  07        CONNOGD TPA
E87F  4C                INCA
E880  20 8F             BRA SETUS
                :
                :
```

```
E882  4C        CTABLE  FCC 'L'
E883  E4 2E             FDB LOAD
E885  47                FCC 'G'
E886  E4 20             FDB GO
E888  50                FCC 'P'
E889  E5 2E             FDB PUNCH
E88B  4D                FCC 'M'
E88C  E6 39             FDB MOVE
E88E  53                FCC 'S'
E88F  E5 F4             FDB SM
E891  44                FCC 'D'
E892  E3 4C             FDB DISMEM
E894  52                FCC 'R'
E895  E2 C5             FDB PREGS
E897  45                FCC 'E'
E898  E3 E2             FDB EOF
E89A  43                FCC 'C'
E89B  BF 05             FDB COLD
E89D  57                FCC 'W'
E89E  BF 0B             FDB WARM
E8A0  4F                FCC 'O'
E8A1  BF 0B             FDB ODESSEY
E8A3  42        QBADR   FCB 42H
E8A4  41                FCB 41H
E8A5  44                FCB 44H
E8A6  20                FCB 20H
E8A7  41                FCB 41H
E8A8  44                FCB 44H
E8A9  52                FCB 52H
E8AA  04                FCB 04
E8AB  52        QRNGERR FCB 52H
E8AC  41                FCB 41H
E8AD  4E                FCB 4EH
E8AE  47                FCB 47H
E8AF  45                FCB 45H
E8B0  20                FCB 20H
E8B1  45                FCB 45H
E8B2  52                FCB 52H
E8B3  52                FCB 52H
E8B4  04                FCB 04
E8B5  45        QEOF    FCB 45H
E8B6  4F                FCB 4FH
E8B7  46                FCB 46H
E8B8  04                FCB 04
E8B9  3F        QQUES   FCB 3FH
E8BA  3F                FCB 3FH
E8BB  3F                FCB 3FH
E8BC  3F                FCB 3FH
E8BD  04                FCB 04
E8BE  43        QCKSMER FCB 43H
E8BF  4B                FCB 4BH
E8C0  53                FCB 53H
```

```
E8C1  4D                   FCB 4DH
E8C2  20                   FCB 20H
E8C3  45                   FCB 45H
E8C4  52                   FCB 52H
E8C5  52                   FCB 52H
E8C6  04                   FCB 04
E8C7  54          QTAPER   FCB 54H
E8C8  41                   FCB 41H
E8C9  50                   FCB 50H
E8CA  45                   FCB 45H
E8CB  20                   FCB 20H
E8CC  45                   FCB 45H
E8CD  52                   FCB 52H
E8CE  52                   FCB 52H
E8CF  04                   FCB 04
E8D0  53          QTPEOF   FCB 53H
E8D1  39                   FCB 39H
E8D2  30                   FCB 30H
E8D3  33                   FCB 33H
E8D4  30                   FCB 30H
E8D5  30                   FCB 30H
E8D6  30                   FCB 30H
E8D7  30                   FCB 30H
E8D8  46                   FCB 46H
E8D9  43                   FCB 43H
E8DA  04                   FCB 04
E8DB  0D          QSI      FCB 0DH
E8DC  0A                   FCB 0AH
E8DD  00                   FCB 00
E8DE  00                   FCB 00
E8DF  00                   FCB 00
E8E0  00                   FCB 00
E8E1  53                   FCB 53H
E8E2  31                   FCB 31H
E8E3  04                   FCB 04
E8E4  20 20 20 20 Q6SPACE FCC '      '
E8E8  20 20
E8EA  04                   FCB 04H
E8EB  20 43 20 20 QREGNAM FCC ' C  B  A  X    P    S'
E8EF  42 20 20 41
E8F3  20 20 20 58
E8F7  20 20 20 20
E8FB  20 50 20 20
E8FF  20 20 20 53
E903  04                   FCB 04H
                  ;
                  ;
                  ;*************************************************************
E904              ;        get navinit.a         ;INITIALIZATION OF I/O DEVICES
                  ;
                  ;
                  ;
```

```
                        ;       INTERRUPT HANDELER FOR TIME CLOCK AND SONAR AT 10 HZ
                        ;       THIS SUBROUTINE IS CALLED BY HARDWARE INTERRUPT (IRQ) AND AS
                        ;       SUCH, ALL REGISTERS AND SYSTEM STATUS ARE PRESERVED ON THE STACK.
                        ;       ON ENTRY, THE PROCESSOR WILL SET THE INTERRUPT MASK.  IT WILL
                        ;       CLEAR THE MASK ON EXIT AND RESTORE THE PROCESSOR REGISTERS/STATUS
                        ;       ON EXIT.
                        ;
E904    BD E9 1E        TICHAN   JSR    FEE         ;GO CHECK OPTICAL SHAFT ENCODERS FIRST
E907    3B                       RTI
                        ;
                        ;
                        ;
E908    6C 00           INCMEM  INC X         ;Increment memory contents
E90A    26 04                   BNE INRET     ;If result not 0, return
E90C    09                      DEX           ;Advance meory pointer
E90D    5A                      DECB          ;Decrement precision counter
E90E    26 F8                   BNE INCMEM    ;If counter <>0, continue incrementing
E910    39              INRET   RTS           ;Return
                        ;
                        ;
E911    A6 00           DCRMEM  LDAA X        ;Fetch byte from memory
E913    80 01                   SUBA 001H     ;Decrement the byte
E915    A7 00                   STAA X        ;Save byte in memory
E917    24 04                   BCC DCRET     ;If no borrow required, return
E919    09                      DEX           ;Advance memory pointer
E91A    5A                      DECB          ;Decrement precision counter
E91B    26 F4                   BNE DCRMEM    ;If counter <>0, continue decrementing
E91D    39              DCRET   RTS           ;Return
                        ;
                        ;
                        ;**************************************************************
                        ;*                                                            *
                        ;*                                                            *
                        ;*      DATE:   8 OCT 84                                      *
                        ;*      VERSION: 1.0                                          *
                        ;*      NAME: FEE      (AS IN THE NURSERY RHYME)              *
                        ;*      FUNCTION: THIS INTERRUPT HANDLER UPDATES THE COUNTS    *
                        ;*                IN PREDEFINED INCREMENTS (DEFAULT 1" (3FH)  *
                        ;*                WHEN THE PTM REACHES ITS TIME-OUT (I.E.     *
                        ;*                INCREMENT DISTANCE HAS BEEN MEASURED, THEN  *
                        ;*                THE PTM AUTOMATICALLY RESETS THE COUNTER    *
                        ;*                AND THIS ROUTINE INCREMENTS THE COUNT IN    *
                        ;*                MEMORY                                      *
                        ;*                THE FRONT WHEEL OSE MEASURES THE ANGULAR    *
                        ;*                POSITION OF THE FRONT WHEEL AND NOT DISTANCE *
                        ;*                TRAVELED BY THE FRONT WHEEL.                *
                        ;*                                                            *
                        ;*      OUTPUTS: INCREMENTS THE FOLLOWING REGISTERS IN MEMORY: *
                        ;*               FWCOUNT                                      *
                        ;*               RWCOUNT1, RWCOUNT2, RWMOD1, RWMOD2,          *
                        ;*               LWCOUNT1, LWCOUNT2, LWMOD1, LWMOD2           *
                        ;**************************************************************
```

```
                          ;
                          ;
                          ;
E91E  B6 C0 31            FEE         LDAA   IO+PTMF+PTMSTAT      ;GET FRONT WHEEL STATUS
E921  48                              ASLA                        ;CHECK FRONT WHEEL IRQ FLAG
E922  25 0F                           BCS    FRONTWHEEL           ;CONTINUE IF TRUE INTERRUPT
E924  B6 C0 41            FIE         LDAA   IO+PTMR+PTMSTAT      ;ELSE GET RIGHT WHEEL STATUS
E927  48                              ASLA                        ;CHECK RIGHT WHEEL IRQ FLAG
E928  25 1E                           BCS    RIGHTWHEEL           ;CONTINUE IF TRUE INTERRUPT
E92A  B6 C0 39            FOE         LDAA   IO+PTML+PTMSTAT      ;ELSE GET LEFT WHEEL STATUS
E92D  48                              ASLA                        ;CHECK LEFT WHEEL IRQ FLAG
E92E  25 37                           BCS    LEFTWHEEL            ;CONTINUE IF TRUE INTERRUPT
E930  7E E9 C2            FUM         JMP    TICHAN1              ;DONE WITH OSE INTERRUPTS
                          ;I
                          ;  HEAR
                          ;       THE
                          ;           CHIRP
                          ;                OF
                          ;                   SONAR
                          ;                         ONE!
                          ;      (SO GO CHECK THE SONAR INTERRUPTS)
                          ;
                          ;
E933  47                  FRONTWHEEL  ASRA                        ;
E934  47                              ASRA                        ;SEE IF COUNTER 1 GENERATED IR..
E935  24 06                           BCC    CHKFWC2              ;IF NOT THEN SERVICE COUNTER 2
E937  7A BE F8            CHKFWC1     DEC    FWCOUNT+1            ;WHEEL HAS TURNED CCW
E93A  FE C0 32                        LDX    IO+PTMF+T.CNT1       ;CLEAR INTERRUPT
E93D  47                  CHKFWC2     ASRA                        ;SEE IF COUNTER 2 GENERATED IRQ
E93E  24 06                           BCC    FWDONE               ;IF NOT CHECK OTHER WHEELS
E940  7C BE F8                        INC    FWCOUNT+1            ;WHEEL HAS TURNED CLOCKWISE
E943  FE C0 34                        LDX    IO+PTMF+T.CNT2       ;CLEAR INTERRUPT
E946  20 DC               FWDONE      BRA    FIE                  ;GO CHECK OTHER WHEELS
                          ;
E948  47                  RIGHTWHEEL  ASRA                        ;
E949  47                              ASRA                        ;SEE IF COUNTER 1 GENERATED IRQ
E94A  24 0B                           BCC    CHKRWC2              ;IF NOT THEN SERVICE COUNTER 2
E94C  C6 02               CHKRWC1     LDAB   #02                  ;#BYTES IN RWCOUNT1 (SIZE)
E94E  CE BE FA                        LDX    #RWCOUNT1+1          ;SET POINTER TO RWCOUNT1 LSBYTE
E951  BD E9 08                        JSR    INCMEM               ;WHEEL HAS MOVED FORWARD 1"
                                                                  ;THAT'S ONE SMALL STEP FOR MARRS-1
                                                                  ;ONE GIANT LEAP FOR ROBOT KIND!!
E954  FE C0 42                        LDX    IO+PTMR+T.CNT1       ;CLEAR INTERRUPT
E957  47                  CHKRWC2     ASRA                        ;SEE IF COUNTER 2 GENERATED IRQ
E958  24 0B                           BCC    RWDONE               ;IF NOT CHECK OTHER WHEELS
E95A  C6 02                           LDAB   #02                  ;#BYTES IN RWCOUNT2 (SIZE)
E95C  CE BE FC                        LDX    #RWCOUNT2+1          ;SET POINTER TO RWCOUNT2 LSBYTE
E95F  BD E9 08                        JSR    INCMEM               ;WHEEL HAS MOVED BACKWARDS
E962  FE C0 44                        LDX    IO+PTMR+T.CNT2       ;CLEAR INTERRUPT
E965  20 C3               RWDONE      BRA    FOE                  ;GO CHECK OTHER WHEELS
                          ;
E967  47                  LEFTWHEEL   ASRA                        ;
```

```
E968  47                        ASRA                        ;SEE IF COUNTER 1 GENERATED IRQ
E969  24 0B                     BCC     CHKLWC2             ;IF NOT THEN SERVICE COUNTER 2
E96B  C6 02         CHKLWC1     LDAB    #02                 ;#BYTES IN LWCOUNT1 (SIZE)
E96D  CE BE FE                  LDX     #LWCOUNT1+1         ;SET POINTER TO LWCOUNT1 LSBYTE
E970  BD E9 08                  JSR     INCMEM              ;WHEEL HAS MOVED FORWARD 1"
E973  FE C0 3A                  LDX     IO+PTML+T.CNT1      ;CLEAR INTERRUPT
E976  47            CHKLWC2     ASRA                        ;SEE IF COUNTER 2 GENERATED IRQ
E977  24 0B                     BCC     LWDONE              ;IF NOT CHECK OTHER WHEELS
E979  C6 02                     LDAB    #02                 ;#BYTES IN LWCOUNT2 (SIZE)
E97B  CE BF 00                  LDX     #LWCOUNT2+1         ;SET POINTER TO LWCOUNT2 LSBYTE
E97E  BD E9 08                  JSR     INCMEM              ;WHEEL HAS MOVED BACKWARDS
E981  FE C0 3C                  LDX     IO+PTML+T.CNT2      ;CLEAR INTERRUPT
E984  20 AA         LWDONE      BRA     FUN                 ;RETURN TO MAIN INTERRUPT HANDLER
                    ;
                    ;*************************************************************
                    ;*                                                           *
                    ;*                                                           *
                    ;*      DATE:    8 Oct 84                                    *
                    ;*      VERSION: 1.0                                         *
                    ;*      NAME: OSEPRELOAD                                     *
                    ;*      MODULE NUMBER:                                       *
                    ;*      FUNCTION: LOADS VALUES YOU SPECIFY INTO PTM LATCHES  *
                    ;*                OF MARRS-1 OPTICAL SHAFT ENCODER PTM'S     *
                    ;*      INPUTS: PRELOAD VALUES (16 BIT) MUST BE IN X REGISTER *
                    ;*              USE THE FOLLOWING FORMULA TO CONVERT FROM    *
                    ;*              INCHES TO COUNTS:                            *
                    ;*                                                           *
                    ;*                    COUNTS = ($40 * DISTANCE (INCH) )  -1  *
                    ;*              THIS IS FOR THE LEFT AND RIGHT WHEELS        *
                    ;*            NOTE: EACH WHEEL COUNT PRELOAD ROUTINE IS A    *
                    ;*                  SEPERATE CALLABLE ROUTINE                *
                    ;*                                                           *
                    ;*      OUTPUTS: THE FOLLOWING MEMORY REGISTERS ARE AFFECTED: *
                    ;*               BCPTMF1, BCPTMF2, BCPTMR1, BCPTMR2, BCPTML2, *
                    ;*               AND BCPTML2                                 *
                    ;*                                                           *
                    ;*                                                           *
                    ;*      HISTORY:                                             *
                    ;*                                                           *
                    ;*************************************************************
                    ;
                    ;
                    ;
                    ;OSE PRELOAD ROUTINES
                    ;
E986  FF BE AB      PRELPTMF    STX     BCPTMF1         ;CLOCKWISE ROTATION
E989  FF BE AD                  STX     BCPTMF2         ;COUNTERCLOCKWISE ROTATION
E98C  BD EC 27                  JSR     INTPTMF         ;RE-INITIALIZE FRONT WHEEL
                                                        ;WITH NEW VALUES
E98F  39                        RTS                     ;RETURN TO CALLER
E990  FF BE B7      PRELPTMR    STX     BCPTMR1         ;FORWARD MOTION
E993  FF BE B9                  STX     BCPTMR2         ;REVERSE MOTION
```

```
E996  BD EC 7F                  JSR    INTPTMR         ;RE-INITIALIZE RIGHT WHEEL
E999  39                        RTS                    ;RETURN TO CALLER
E99A  FF BE B1     PRELPTML     STX    BCPTML1         ;FORWARD MOTION
E99D  FF BE B3                  STX    BCPTML2         ;REVERSE MOTION
E9A0  BD EC 53                  JSR    INTPTML         ;RE-INITIALIZE LEFT WHEEL
E9A3  39                        RTS                    ;RETURN TO CALLER
                   ;
                   ;*****************************************************
                   ;*                                                   *
                   ;*                                                   *
                   ;*      DATE:   8 OCT 84                             *
                   ;*      VERSION: 1.0                                 *
                   ;*      NAME: FWRESET, RWRESET, AND LWRESET          *
                   ;*                                                   *
                   ;*      FUNCTION: ALLOWS USER TO RESET WHEEL COUNT VALUES *
                   ;*                TO ANY VALUE HE DESIRES WITHIN THE *
                   ;*                FOLLOWING LIMITS:                  *
                   ;*                FWCOUNT  MAX 599 DECIMAL           *
                   ;*                RWCOUNT1, RWCOUNT2, LWCOUNT1, AND LWCOUNT2 *
                   ;*                ALL CAN HAVE MAX COUNTS OF 0FFFFH  *
                   ;*                THIS ROUTINE ALSO CLEARS THE MODULO COUNT *
                   ;*                FOR THE LEFT AND RIGHT WHEELS      *
                   ;*                                                   *
                   ;*      INPUTS:   X REG - NEW VALUE AND CALL THE     *
                   ;*                APPROPRIATE ROUTINE                *
                   ;*                                                   *
                   ;*                                                   *
                   ;*****************************************************
                   ;
                   ;
E9A4  FF BE F7     FWRESET      STX    FWCOUNT
E9A7  39                        RTS
E9A8  FF BE F9     RWRESET      STX    RWCOUNT1
E9AB  FF BE FB                  STX    RWCOUNT2
E9AE  7F BF 01                  CLR    RWMOD1
E9B1  7F BF 02                  CLR    RWMOD2
E9B4  39                        RTS
E9B5  FF BE FD     LWRESET      STX    LWCOUNT1
E9B8  FF BE FF                  STX    LWCOUNT2
E9BB  7F BF 03                  CLR    LWMOD1
E9BE  7F BF 04                  CLR    LWMOD2
E9C1  39                        RTS
                   ;
                   ;
                   ;
                   ;   COMMENTS CONCERNING THE PTM'S FOR THE WHEEL OPTICAL SHAFT ENCODERS
                   ;
                   ;   CIRCUMFERENCE OF LEFT AND RIGHT WHEELS IS 18.75"
                   ;   THERE ARE 1200 COUNTS PER REVOLUTION AVAILABLE WITH THE
                   ;   DATAMETRICS K-3 OPTICAL SHAFT ENCODER.   THUS THERE ARE
                   ;   64 COUNTS (3FH) PER 1" DISTANCE TRAVELED
                   ;
```

```
                         ;
                         ;  RIGHT WHEEL:
                         ;     FORWARD MOTION
                         ;     A CHANNEL (COUNTER 2) GATES B CHANNEL (COUNTER 1)
                         ;
                         ;     REVERSE MOTION
                         ;     B CHANNEL (COUNTER 1) GATES A CHANNEL (COUNTER 2)
                         ;
                         ;  LEFT WHEEL:
                         ;     FORWARD MOTION
                         ;     B CHANNEL (COUNTER 2) GATES A CHANNEL (COUNTER 1)
                         ;
                         ;     REVERSE MOTION
                         ;     A CHANNEL (COUNTER 1) GATES B CHANNEL (COUNTER 2)
                         ;
                         ;     FRONT WHEEL PTM KEEPS TRACK OF ABSOLUTE HEADING
                         ;     OF FRONT DRIVE WHEEL.  WITH THE DRIVE WHEEL CENTERED,
                         ;     THE COUNT AT FWCOUNT+1 SHOULD BE $49.
                         ;     WITH THE FRONT DRIVE WHEEL MAX LEFT FWCOUNT+1 SHOULD BE $FF.
                         ;     WITH THE FRONT DRIVE WHEEL 90 DEGREES RIGHT OF CENTER, THE FWCOUNT+1
                         ;     SHOULD BE $94.
                         ;     THIS IS ASSUMING THAT A DEFAULT VALUE OF $03 IS USED
                         ;     AS AN INCREMENTAL VALUE ON THE FRONT WHEEL PTM ALSO
                         ;     KNOWN AS BCPTMF1 OR BCPTMF2
                         ;
                         ;  FRONT WHEEL:
                         ;
                         ;     CLOCKWISE MOTION
                         ;     B CHANNEL (COUNTER 2) GATES A CHANNEL (COUNTER 1)
                         ;
                         ;     COUNTER-CLOCKWISE MOTION
                         ;     A CHANNEL (COUNTER 1) GATES B CHANNEL (COUNTER 2)
                         ;
                         ;
                         ;
                         ;
E9C2   CE C0 00          TICHAN1  LDX    #IO              ;POINT TO I/O AREA
E9C5   A6 49                      LDAA   PIAAB+P.SRA,X    ;GET PIA A STATUS
E9C7   48                         ASLA                    ;CHECK INTERRUPT FLAG OF PIA A
E9C8   25 03                      BCS    TICHAN2          ;CONTINUE IF TRUE INTERRUPT
E9CA   7E BE C7                   JMP    MORETIC          ;ELSE RETURN TO CALLER
                         ;
E9CD   EE 1A             TICHAN2  LDX    PTMAB+T.CNT1,X   ;GET SONAR READING FROM CHANNEL A
E9CF   FF BE E6                   STX    TICTEMP0         ;SAVE IT
E9D2   B6 BE E7                   LDAA   TICTEMP0+1       ;GET LOW 8 BITS
E9D5   43                         COMA                    ;DATA IS IN NEGATIVE 1'S COMPLEMENT
                                                          ;SO WE MUST CONVERT IT TO BE POSITIVE
                                                          ;THE DISTANCE MEASURED
                                                          ;IS FROM THE CENTER OF THE ROBOT
E9D6   B7 BE C1                   STAA   SONARA
                         ;
E9D9   CE C0 00                   LDX    #IO              ;POINT TO I/O AREA
```

```
E9DC   EE 1C              LDX   PTMAB+T.CNT2,X  ;GET SONAR READING FROM CHANNEL B
E9DE   FF BE E6           STX   TICTEMP0        ;SAVE IT
E9E1   B6 BE E7           LDAA  TICTEMP0+1      ;GET LOW 8 BITS
E9E4   43                 COMA                  ;DATA IS IN ONE'S COMPLEMENT
                                                ;SO WE MUST CONVERT IT TO BE POSITIVE
                                                ;THE DISTANCE MEASURED
                                                ;IS FROM THE CENTER OF THE ROBOT
E9E5   B7 BE C2           STAA  SONARB
                   ;
E9E8   CE C0 00           LDX   #IO             ;POINT TO I/O AREA
E9EB   EE 22              LDX   PTMCD+T.CNT1,X  ;GET SONAR READING FROM CHANNEL C
E9ED   FF BE E6           STX   TICTEMP0        ;SAVE IT
E9F0   B6 BE E7           LDAA  TICTEMP0+1      ;GET LOW 8 BITS
E9F3   43                 COMA                  ;DATA IS IN ONE'S COMPLEMENT
                                                ;SO WE MUST CONVERT IT TO BE POSITIVE
                                                ;THE DISTANCE MEASURED
                                                ;IS FROM THE CENTER OF THE ROBOT
E9F4   B7 BE C3           STAA  SONARC
                   ;
E9F7   CE C0 00           LDX   #IO             ;POINT TO I/O AREA
E9FA   EE 24              LDX   PTMCD+T.CNT2,X  ;GET SONAR READING FROM CHANNEL D
E9FC   FF BE E6           STX   TICTEMP0        ;SAVE IT
E9FF   B6 BE E7           LDAA  TICTEMP0+1      ;GET LOW 8 BITS
EA02   43                 COMA                  ;DATA IS IN ONE'S COMPLEMENT
                                                ;SO WE MUST CONVERT IT TO BE POSITIVE
                                                ;THE DISTANCE MEASURED
                                                ;IS FROM THE CENTER OF THE ROBOT
EA03   B7 BE C4           STAA  SONARD
                   ;
EA06   FE C0 2E           LDX   IO+PTMV+T.CNT3  ;UPDATE TIME SINCE TIME ZERO
EA09   08                 INX
EA0A   FF BE D9           STX   TICTIME         ;TIME FROM COUNTER IS IN ONE'S COMPLEMENT
EA0D   73 BE D9           COM   TICTIME
EA10   73 BE DA           COM   TICTIME+1
EA13   FE BE D9           LDX   TICTIME
EA16   FF BE C5           STX   SONTIME         ;UPDATE TIME OF LAST SONAR READING
EA19   CE C0 00           LDX   #IO             ;POINT TO IO AREA
EA1C   A6 48              LDAA  PIAAB+PIAPA,X   ;GET SONAR SELECTED FOR A CHANNEL
EA1E   B7 BE BD           STAA  SONARSLA        ;TELL USER WHICH SONAR WAS SELECTED
EA21   A6 4A              LDAA  PIAAB+PIAPB,X   ;GET SONAR SELECTED FOR B CHANNEL
EA23   B7 BE BE           STAA  SONARSLB        ;TELL USER WHICH SONAR WAS SELECTED
EA26   A6 50              LDAA  PIACD+PIAPC,X   ;GET SONAR SELECTED FOR C CHANNEL
EA28   B7 BE BF           STAA  SONARSLC        ;TELL USER WHICH SONAR WAS SELECTED
EA2B   A6 52              LDAA  PIACD+PIAPD,X   ;GET SONAR SELECTED FOR D CHANNEL
EA2D   B7 BE C0           STAA  SONARSLD        ;TELL USER WHICH SONAR WAS SELECTED
EA30   7D BE E1           TST   SONCHANGE       ;SEE IF NEW SONARS ARE TO BE SELECTED
EA33   27 03              BEQ   TICHAN3
EA35   7E BE C7           JMP   MORETIC
EA38   B6 BE E2   TICHAN3 LDAA  SONNEXA         ;GET NEXT SONAR SELECT BYTE FOR SONAR A
EA3B   A7 48              STAA  PIAAB+PIAPA,X   ;WRITE IT OUT TO THE PIA
EA3D   B6 BE E3           LDAA  SONNEXB         ;GET NEXT SONAR SELECT BYTE FOR SONAR B
EA40   A7 4A              STAA  PIAAB+PIAPB,X   ;WRITE IT OUT TO THE PIA
```

```
EA42  B6 BE E4            LDAA  SONNEXC          ;GET NEXT SONAR SELECT BYTE FOR SONAR C
EA45  A7 50               STAA  PIACD+PIAPC,X    ;WRITE IT OUT TO THE PIA
EA47  B6 BE E5            LDAA  SONNEXD          ;GET NEXT SONAR SELECT BYTE FOR SONAR D
EA4A  A7 52               STAA  PIACD+PIAPD,X    ;WRITE IT OUT TO THE PIA
EA4C  86 55               LDAA  #055H
EA4E  B7 BE E1            STAA  SONCHANGE        ;TELL USER SONAR SELECTS HAVE BEEN CHANGED
EA51  7E BE C7            JMP   MORETIC
                     ;
                     ;
                     ;
                     ;
                     ;    SUBROUTINE:    READSONAR
                     ;                   BUFFERS USER PROGRAM FROM INTERRUPT PROCESSING
                     ;                   OF SONAR DATA.  SONAR SELECT-READING PAIRS IN
                     ;                   LOCATIONS SONDATSA THRU SONDATRD ARE PLACED THERE
                     ;                   BY THIS SUBROUTINE AND AS SUCH ARE GUARENTEED TO
                     ;                   REPRESENT VALID/CURRENT SONAR READINGS.
                     ;
                     ;    ################################################
                     ;
                     ;          P L E A S E  T A K E  N O T E  ! ! ! ! ! ! !
                     ;
                     ;    ################################################
                     ;
                     ;          SONAR DATA READINGS SHOULD BE OBTAINED BY USER PROGRAMS
                     ;          BY CALLING THIS SUBROUTINE AND THEN CHECKING THE VALUES
                     ;          STORED IN LOCATIONS SONDATSA THRU SONDATRD I.E.:
                     ;
                     ;                   SONDATSA    THIS IS THE SELECT BYTE FOR SONAR A
                     ;
                     ;                   SONDATRA    THIS IS THE READING FROM SONAR A
                     ;
                     ;                   SONDATSB    THIS IS THE SELECT BYTE FOR SONAR B
                     ;
                     ;                   SONDATRB    THIS IS THE READING FROM SONAR B
                     ;
                     ;                   SONDATSC    THIS IS THE SELECT BYTE FOR SONAR C
                     ;
                     ;                   SONDATRC    THIS IS THE READING FROM SONAR C
                     ;
                     ;                   SONDATSD    THIS IS THE SELECT BYTE FOR SONAR D
                     ;
                     ;                   SONDATRD    THIS IS THE READING FROM SONAR D
                     ;
                     ;    ######################################################
                     ;
                     ;
EA54  0F             READSONAR SEI                  ;MASK THE INTERRUPT SYSTEM
EA55  36                       PSHA                 ;SAVE THE A REGISTER
EA56  B6 BE BD                 LDAA  SSLA           ;GET SONAR SELECT BYTE A
EA59  B7 BE EE                 STAA  SONDATSA       ;STORE SONAR SELECT BYTE A
EA5C  B6 BE BE                 LDAA  SSLB           ;GET SONAR SELECT BYTE B
```

```
EA5F   B7 BE F1             STAA  SONDATSB      ;STORE SONAR SELECT BYTE B
EA62   B6 BE BF             LDAA  SSLC          ;GET SONAR SELECT BYTE C
EA65   B7 BE F3             STAA  SONDATSC      ;STORE SONAR SELECT BYTE C
EA68   B6 BE C0             LDAA  SSLD          ;GET SONAR SELECT BYTE D
EA6B   B7 BE F5             STAA  SONDATSD      ;STORE SONAR SELECT BYTE D
EA6E   B6 BE C1             LDAA  SONARA        ;GET SONAR READING FOR SONAR A
EA71   B7 BE F0             STAA  SONDATRA      ;STORE SONAR READING FOR SONAR A
EA74   B6 BE C2             LDAA  SONARB        ;GET SONAR READING FOR SONAR B
EA77   B7 BE F2             STAA  SONDATRB      ;STORE SONAR READING FOR SONAR B
EA7A   B6 BE C3             LDAA  SONARC        ;GET SONAR READING FOR SONAR C
EA7D   B7 BE F4             STAA  SONDATRC      ;STORE SONAR READING FOR SONAR C
EA80   B6 BE C4             LDAA  SONARD        ;GET SONAR READING FOR SONAR D
EA83   B7 BE F6             STAA  SONDATRD      ;STORE SONAR READING FOR SONAR D
EA86   7D BE EE             TST   INTMASK       ;CHECK TO SEE IF OTHERS WANT INTERRUPTS MASKED
EA89   26 01                BNE   RDSONDONE     ;DONE IF YES
EA8B   0E                   CLI                 ;ELSE CLEAR INTERRUPT MASK
EA8C   32         RDSONDONE PULA                ;RESTORE A REGISTER
EA8D   39                   RTS                 ;RETURN TO CALLER
                  ;
                  ;
                  ;
                  ;       SUBROUTINE POS (PUSH ON STACK) IS REENTRANT BUT MAY NOT
                  ;       BE USED BY AN INTERRUPT ROUTINE SINCE THE SYSTEM TEMPORARY
                  ;       REGISTERS ARE USED.
                  ;       POS PUSHES N BYTES ONTO THE STACK.  ON ENTRY, THE A REGISTER
                  ;       CONTAINS THE BYTE COUNT (N) AND THE X REGISTER POINTS TO THE
                  ;       FIRST BYTE TO BE PUSHED ON TO THE STACK.  THE B AND C REGISTERS
                  ;       ARE CLOBBERED.  ON RETURN, X = X + N + 1,  A = N, SP = SP - (N + 1)
                  ;       BECAUSE THE STACK GROWS DOWNWARD!  THE STACK POINTER MUST EVENTUALLY
                  ;       RETURN TO IT'S PREVIOUS VALUE OR OTHER SYSTEM PROGRAMS OR DATA WILL
                  ;       BE CLOBBERED BY THE DOWNWARD GROWING STACK!
                  ;
EA8E   0F         POS      SEI                  ;MASK HARDWARE INTERRUPTS
EA8F   B7 BF D5            STAA   TEMP0         ;SAVE THE A REGISTER
EA92   FF BF D7            STX    TEMP1         ;AND THE X REGISTER
EA95   30                  TSX                  ;GET THE RETURN ADDRESS OF THE SUBROUTINE
                                                ;CALLER
EA96   EE 00               LDX    0,X
EA98   FF BF D9            STX    TEMP2         ;SAVE THE RETURN ADDRESS
EA9B   FE BF D7            LDX    TEMP1         ;POINT TO THE BYTES TO BE PUSHED ON THE STACK
EA9E   16                  TAB                  ;BYTE COUNT IS NOW IN B REGISTER
                  ;
EA9F   A6 00      LOOPPOS1 LDAA   0,X           ;LOOP HERE N TIMES
EAA1   36                  PSHA                 ;PUSHING VALUES ONTO THE STACK
EAA2   08                  INX                  ;POINT TO NEXT BYTE
EAA3   5A                  DECB                 ;ONE LESS BYTE TO PUSH
EAA4   26 F9               BNE    LOOPPOS1      ;IF B IS NOT ZERO, LOOP UNTIL DONE
                  ;
EAA6   B6 BF DA            LDAA   TEMP2+1       ;ELSE PUSH RETURN ADDRESS
EAA9   36                  PSHA                 ;ONTO THE STACK
EAAA   B6 BF D9            LDAA   TEMP2
EAAD   36                  PSHA
```

```
EAAE  B6 BF D5          LDAA  TEMP0       ;RESTORE THE BYTE COUNT TO THE A REGISTER
EAB1  F6 BE EE          LDAB  INTMASK     ;CHECK THE INTERRUPT MASK BYTE
EAB4  26 01             BNE   POSDONE     ;IF NOT ZERO WE ARE DONE
EAB6  0E                CLI               ;ELSE CLEAR THE INTERRUPT FLAG AND
EAB7  39       POSDONE  RTS               ;RETURN TO CALLER
               ;
               ;
               ;
               ;
               ;        SUBROUTINE: SIZEMEM   SIZES MEMORY.  IF PIAAB BIT CA-2 IS HIGH,
               ;                              THEN DATA IN ROMS SHAREING ADDRESS SPACE
               ;                              WITH RAMS WILL BE LOADED INTO THE RAMS AS
               ;                              A SIDE EFFECT. RAMS WILL APPEAR AS ROMS
               ;                              UNTIL CA-2 IS SET LOW.  USE SUBROUTINE
               ;                              ROMLAYRAM FOR AUTOMATIC ROM TO RAM OVERLAY
               ;                              AND CA-2 SET TO LOW.
               ;
EAB8  0F       SIZEMEM  SEI               ;SET INTERRUPT MASK
EAB9  7F BF D5          CLR   TEMP0       ;CLEAR GOOD MEMORY FLAG ($00=NOT GOOD,
                                          ; $FF=GOOD)
EABC  FE BE DD          LDX   RAMSIZEL    ;POINT TO LOW RAM ADDRESS
               ;
EABF  A6 00    SIZELP1  LDAA  0,X         ;SAVE VALUE OF MEMORY ADDRESS IN A REGISTER
EAC1  C6 55             LDAB  #055H       ;START MEMORY TEST TO SEE IF LOCATION IS GOOD
EAC3  E7 00             STAB  0,X
EAC5  E6 00             LDAB  0,X
EAC7  C1 55             CMPB  #055H
EAC9  26 19             BNE   BADSZMEM
EACB  C6 AA             LDAB  #0AAH
EACD  E7 00             STAB  0,X
EACF  E6 00             LDAB  0,X
EAD1  C1 AA             CMPB  #0AAH
EAD3  26 0F             BNE   BADSZMEM
EAD5  A7 00             STAA  0,X         ;RESTORE RAM BYTE AS IT WAS BEFORE
EAD7  7D BF D5          TST   TEMP0       ;IS THIS THE FIRST GOOD BYTE?
EADA  27 15             BEQ   FSTSZGD     ;BRANCH IF YES
EADC  08                INX               ;ELSE POINT TO NEXT MEMORY LOCATION
EADD  BC BE DF          CPX   RAMSIZEH    ;ARE WE AT THE END OF THE RAM SIZE TEST?
EAE0  27 1D             BEQ   DNESZMEM    ;IF YES THEN DONE
EAE2  20 DB             BRA   SIZELP1     ;ELSE CONTINUE MEMORY SIZE LOOP
EAE4  A7 00    BADSZMEM STAA  0,X         ;RESTORE RAM BYTE AS IT WAS
                                          ;(ROM OVERLAY TO RAM OCCURS HERE IF
                                          ;PIAA CA-2 IS HIGH!)
EAE6  7F BF D5          CLR   TEMP0       ;CLEAR GOOD MEMORY FLAG
EAE9  08                INX               ;POINT TO NEXT MEMORY LOCATION
EAEA  BC BE DF          CPX   RAMSIZEH    ;ARE WE AT THE END OF THE RAM SIZE TEST?
EAED  27 10             BEQ   DNESZMEM    ;IF YES THEN DONE
EAEF  20 CE             BRA   SIZELP1     ;CONTINUE MEMORY SIZE LOOP
EAF1  FF BE DD FSTSZGD  STX   RAMSIZEL    ;SAVE ADDRESS OF FIRST GOOD RAM
EAF4  73 BF D5          COM   TEMP0       ;SET GOOD MEMORY FLAG
EAF7  08                INX               ;POINT TO NEXT MEMORY LOCATION
EAF8  BC BE DF          CPX   RAMSIZEH    ;ARE WE AT THE END OF THE RAM SIZE TEST?
```


131

```
EAFB  27 02            BEQ    DNESZMEM       ;IF YES THEN DONE
EAFD  20 C0            BRA    SIZELP1        ;CONTINUE MEMORY SIZE LOOP
EAFF  09      DNESZMEM DEX
EB00  FF BE DF         STX    RAMSIZEH       ;STORE ADDRESS OF LAST MEMORY LOCATION TESTED
EB03  B6 BE EE         LDAA   INTMASK        ;CHECK TO SEE IF MASK IS TO BE LEFT SET
EB06  2; 01            BNE    DSZMEM2        ;RETURN TO CALLER IF MASK IS TO BE LEFT SET
EB08  0E               CLI                   ;ELSE CLEAR MASK AND
EB09  39      DSZMEM2  RTS                   ;RETURN TO CALLER
              ;
              ;
              ;
              ;
              ;         SUBROUTINE:  ROMLAYRAM        OVERLAYS ROM TO RAM AND RETURNS
              ;                                       RAM TO TRUE RAM STATE OVER ENTIRE
              ;                                       PROCESSOR ADDRESS SPACE
              ;
EB0A  CE 00 00  ROMLAYRAM    LDX   #XZERO           ;POINT TO LOW MEMORY
EB0D  FF BE DD               STX   RAMSIZEL
EB10  CE A0 00               LDX   #0A000H          ;LAST USER RAM + 1
EB13  FF BE DF               STX   RAMSIZEH
EB16  B6 BE 95  ROMOVERRAM   LDAA  CRPIAA           ;ENABLE ROM TO RAM OVERLAY
EB19  8A 38                  ORAA  #038H
EB1B  B7 C0 49               STAA  IO+PIAAB+PIACRA
EB1E  BD EA B8               JSR   SIZEMEM          ;SIZE MEMORY AND OVERLAY ROM TO RAM
EB21  B6 BE 95               LDAA  CRPIAA           ;RESTORE PIA A TO PREVIOUS MODE
EB24  B7 C0 49               STAA  IO+PIAAB+PIACRA  ;THIS DISABLES THE ROM TO RAM OVERLAY
                                                    ;PROCESS AND MAKES RAM LOOK LIKE RAM
EB27  CE 00 00               LDX   #XZERO           ;POINT TO LOW MEMORY
EB2A  FF BE DD               STX   RAMSIZEL
EB2D  CE C0 00               LDX   #IO              ;POINT TO LAST RAM + 1 (I/O AREA)
EB30  FF BE DF               STX   RAMSIZEH
EB33  BD EA B8               JSR   SIZEMEM          ;SIZE MEMORY AGAIN TO SHOW TRUE RAM
EB36  39                     RTS                    ;RETURN TO CALLER
              ;
              ;
              ;
              ;         SUBROUTINE:     INTPIAAB      INITIALIZE PIA AB ACCORDING TO VALUES
              ;                                       STORED IN SYSTEM RAM (COPIES OF THE
              ;                                       CONTROL REGISTERS)
              ;
EB37  CE C0 48  INTPIAAB LDX   #IO+PIAAB      ;POINT TO PIA
EB3A  B6 BE 95           LDAA  CRPIAA         ;GET CONTROL WORD
EB3D  84 FB              ANDA  #0FBH          ;ACCESS DATA DIRECTION REGISTER
EB3F  A7 01              STAA  P.CRA,X
EB41  B6 BE 96           LDAA  CRPIAB
EB44  84 FB              ANDA  #0FBH
EB46  A7 03              STAA  P.CRB,X
EB48  CE FF FF           LDX   #0FFFFH        ;MAKE ALL PIA LINES OUTPUTS
EB4B  FF C0 48           STX   IO+PIAAB+P.DDA
EB4E  FF C0 4A           STX   IO+PIAAB+P.DDB
EB51  CE C0 48           LDX   #IO+PIAAB      ;POINT TO PIA
EB54  B6 BE 95           LDAA  CRPIAA         ;GET CONTROL WORD
```

```
EB57  8A 04                  ORAA   #04H         ;ACCESS PREIPHERAL REGISTER
EB59  A7 01                  STAA   P.CRA,X
EB5B  B6 BE 96               LDAA   CRPIAB
EB5E  8A 04                  ORAA   #04H
EB60  A7 03                  STAA   P.CRB,X
EB62  B6 BE BD               LDAA   SONARSLA     ;SELECT A SONAR
EB65  A7 00                  STAA   P.PRA,X
EB67  B6 BE BE               LDAA   SONARSLB     ;SELECT A SONAR
EB6A  A7 02                  STAA   P.PRB,X
EB6C  39                     RTS                 ;RETURN TO CALLER
                    ;
                    ;
                    ;
                    ;     SUBROUTINE:   INTPIACD      INITIALIZE PIA CD ACCORDING TO VALUES
                    ;                                 STORED IN SYSTEM RAM (COPIES OF THE
                    ;                                 CONTROL REGISTERS)
                    ;
EB6D  CE C0 50      INTPIACD LDX    #IO+PIACD    ;POINT TO PIA
EB70  B6 BE 97               LDAA   CRPIAC       ;GET CONTROL WORD
EB73  84 FB                  ANDA   #0FBH        ;ACCESS DATA DIRECTION REGISTER
EB75  A7 01                  STAA   P.CRC,X
EB77  B6 BE 98               LDAA   CRPIAD
EB7A  84 FB                  ANDA   #0FBH
EB7C  A7 03                  STAA   P.CRD,X
EB7E  CE FF FF               LDX    #0FFFFH      ;MAKE ALL PIA LINES OUTPUTS
EB81  FF C0 50               STX    IO+PIACD+P.DDC
EB84  FF C0 52               STX    IO+PIACD+P.DDD
EB87  CE C0 50               LDX    #IO+PIACD    ;POINT TO PIA
EB8A  B6 BE 97               LDAA   CRPIAC       ;GET CONTROL WORD
EB8D  8A 04                  ORAA   #04H         ;ACCESS PREIPHERAL REGISTER
EB8F  A7 01                  STAA   P.CRC,X
EB91  B6 BE 98               LDAA   CRPIAD
EB94  8A 04                  ORAA   #04H
EB96  A7 03                  STAA   P.CRD,X
EB98  B6 BE BF               LDAA   SONARSLC     ;SELECT A SONAR
EB9B  A7 00                  STAA   P.PRC,X
EB9D  B6 BE C0               LDAA   SONARSLD     ;SELECT A SONAR
EBA0  A7 02                  STAA   P.PRD,X
EBA2  39                     RTS                 ;RETURN TO CALLER
                    ;
                    ;
                    ;
                    ;     SUBROUTINE:   INTPTMAB      INITIALIZE PTM AB FROM VALUES STORED
                    ;                                 IN SYSTEM RAM
                    ;
EBA3  CE C0 18      INTPTMAB LDX    #IO+PTMAB    ;POINT TO PTM AB
EBA6  B6 BE 84               LDAA   CRPTMAB2     ;GET CONTROL WORD FOR TIMER 2
EBA9  A7 01                  STAA   PTMCR2,X
EBAB  B6 BE 85               LDAA   CRPTMAB3     ;GET CONTROL WORD FOR TIMER 3
EBAE  A7 00                  STAA   PTMCR3,X
EBB0  B6 BE 84               LDAA   CRPTMAB2
EBB3  8A 01                  ORAA   #01H         ;POINT TO CONTROL REGISTER 1
```

```
EBB5  A7 01                STAA  PTMCR2,X
EBB7  B6 BE 83             LDAA  CRPTMAB1      ;GET CONTROL WORD FOR TIMER 1
EBBA  A7 00                STAA  PTMCR1,X
EBBC  FE BE 99             LDX   BCPTMAB1      ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EBBF  FF C0 1A             STX   IO+PTMAB+T.LATCH1
EBC2  FE BE 9B             LDX   BCPTMAB2      ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EBC5  FF C0 1C             STX   IO+PTMAB+T.LATCH2
EBC8  FE BE 9D             LDX   BCPTMAB3      ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EBCB  FF C0 1E             STX   IO+PTMAB+T.LATCH3
EBCE  39                   RTS                 ;RETURN TO CALLER
                 ;
                 ;
                 ;
                 ;     SUBROUTINE:   INTPTMCD     INITIALIZE PTM CD FROM VALUES STORED
                 ;                                IN SYSTEM RAM
                 ;
EBCF  CE C0 20   INTPTMCD LDX   #IO+PTMCD      ;POINT TO PTM CD
EBD2  B6 BE 87             LDAA  CRPTMCD2      ;GET CONTROL WORD FOR TIMER 2
EBD5  A7 01                STAA  PTMCR2,X
EBD7  B6 BE 88             LDAA  CRPTMCD3      ;GET CONTROL WORD FOR TIMER 3
EBDA  A7 00                STAA  PTMCR3,X
EBDC  B6 BE 87             LDAA  CRPTMCD2
EBDF  8A 01                ORAA  001H          ;POINT TO CONTROL REGISTER 1
EBE1  A7 01                STAA  PTMCR2,X
EBE3  B6 BE 86             LDAA  CRPTMCD1      ;GET CONTROL WORD FOR TIMER 1
EBE6  A7 00                STAA  PTMCR1,X
EBE8  FE BE 9F             LDX   BCPTMCD1      ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EBEB  FF C0 22             STX   IO+PTMCD+T.LATCH1
EBEE  FE BE A1             LDX   BCPTMCD2      ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EBF1  FF C0 24             STX   IO+PTMCD+T.LATCH2
EBF4  FE BE A3             LDX   BCPTMCD3      ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EBF7  FF C0 26             STX   IO+PTMCD+T.LATCH3
EBFA  39                   RTS                 ;RETURN TO CALLER
                 ;
                 ;
                 ;
                 ;     SUBROUTINE:   INTPTMV      INITIALIZE PTM V FROM VALUES STORED
                 ;                                IN SYSTEM RAM
                 ;
EBFB  CE C0 28   INTPTMV  LDX   #IO+PTMV       ;POINT TO PTM V
EBFE  B6 BE 8A             LDAA  CRPTMV2       ;GET CONTROL WORD FOR TIMER 2
EC01  A7 01                STAA  PTMCR2,X
EC03  B6 BE 8B             LDAA  CRPTMV3       ;GET CONTROL WORD FOR TIMER 3
EC06  A7 00                STAA  PTMCR3,X
EC08  B6 BE 8A             LDAA  CRPTMV2
EC0B  8A 01                ORAA  001H          ;POINT TO CONTROL REGISTER 1
EC0D  A7 01                STAA  PTMCR2,X
EC0F  B6 BE 89             LDAA  CRPTMV1       ;GET CONTROL WORD FOR TIMER 1
EC12  A7 00                STAA  PTMCR1,X
EC14  FE BE A5             LDX   BCPTMV1       ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EC17  FF C0 2A             STX   IO+PTMV+T.LATCH1
EC1A  FE BE A7             LDX   BCPTMV2       ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
```

```
EC1D  FF C0 2C              STX   IO+PTMV+T.LATCH2
EC20  FE BE A9              LDX   BCPTMV3       ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EC23  FF C0 2E              STX   IO+PTMV+T.LATCH3
EC26  39                    RTS                 ;RETURN TO CALLER
                    ;
                    ;
                    ;
                    ;      SUBROUTINE:    INTPTMF        INITIALIZE PTM F FROM VALUES STORED
                    ;                                    IN SYSTEM RAM
                    ;
EC27  CE C0 30      INTPTMF LDX   #IO+PTMF      ;POINT TO PTM F
EC2A  B6 BE 8D              LDAA  CRPTMF2       ;GET CONTROL WORD FOR TIMER 2
EC2D  A7 01                 STAA  PTMCR2,X
EC2F  B6 BE 8E              LDAA  CRPTMF3       ;GET CONTROL WORD FOR TIMER 3
EC32  A7 00                 STAA  PTMCR3,X
EC34  B6 BE 8D              LDAA  CRPTMF2
EC37  8A 01                 ORAA  #01H          ;POINT TO CONTROL REGISTER 1
EC39  A7 01                 STAA  PTMCR2,X
EC3B  B6 BE 8C              LDAA  CRPTMF1       ;GET CONTROL WORD FOR TIMER 1
EC3E  A7 00                 STAA  PTMCR1,X
EC40  FE BE AB              LDX   BCPTMF1       ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EC43  FF C0 32              STX   IO+PTMF+T.LATCH1
EC46  FE BE AD              LDX   BCPTMF2       ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EC49  FF C0 34              STX   IO+PTMF+T.LATCH2
EC4C  FE BE AF              LDX   BCPTMF3       ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EC4F  FF C0 36              STX   IO+PTMF+T.LATCH3
EC52  39                    RTS                 ;RETURN TO CALLER
                    ;
                    ;
                    ;
                    ;      SUBROUTINE:    INTPTML        INITIALIZE PTM L FROM VALUES STORED
                    ;                                    IN SYSTEM RAM
                    ;
EC53  CE C0 38      INTPTML LDX   #IO+PTML      ;POINT TO PTM L
EC56  B6 BE 90              LDAA  CRPTML2       ;GET CONTROL WORD FOR TIMER 2
EC59  A7 01                 STAA  PTMCR2,X
EC5B  B6 BE 91              LDAA  CRPTML3       ;GET CONTROL WORD FOR TIMER 3
EC5E  A7 00                 STAA  PTMCR3,X
EC60  B6 BE 90              LDAA  CRPTML2
EC63  8A 01                 ORAA  #01H          ;POINT TO CONTROL REGISTER 1
EC65  A7 01                 STAA  PTMCR2,X
EC67  B6 BE 8F              LDAA  CRPTML1       ;GET CONTROL WORD FOR TIMER 1
EC6A  A7 00                 STAA  PTMCR1,X
EC6C  FE BE B1              LDX   BCPTML1       ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EC6F  FF C0 3A              STX   IO+PTML+T.LATCH1
EC72  FE BE B3              LDX   BCPTML2       ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EC75  FF C0 3C              STX   IO+PTML+T.LATCH2
EC78  FE BE B5              LDX   BCPTML3       ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EC7B  FF C0 3E              STX   IO+PTML+T.LATCH3
EC7E  39                    RTS                 ;RETURN TO CALLER
                    ;
                    ;
```

```
                            ;
                            ;       SUBROUTINE:     INTPTMR         INITIALIZE PTM R FROM VALUES STORED
                            ;                                       IN SYSTEM RAM
                            ;
EC7F    CE C0 40            INTPTMR LDX     #IO+PTMR        ;POINT TO PTM R
EC82    B6 BE 93                    LDAA    CRPTMR2         ;GET CONTROL WORD FOR TIMER 2
EC85    A7 01                       STAA    PTMCR2,X
EC87    B6 BE 94                    LDAA    CRPTMR3         ;GET CONTROL WORD FOR TIMER 3
EC8A    A7 00                       STAA    PTMCR3,X
EC8C    B6 BE 93                    LDAA    CRPTMR2
EC8F    8A 01                       ORAA    #01H            ;POINT TO CONTROL REGISTER 1
EC91    A7 01                       STAA    PTMCR2,X
EC93    B6 BE 92                    LDAA    CRPTMR1         ;GET CONTROL WORD FOR TIMER 1
EC96    A7 00                       STAA    PTMCR1,X
EC98    FE BE B7                    LDX     BCPTMR1         ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
EC9B    FF C0 42                    STX     IO+PTMR+T.LATCH1
EC9E    FE BE B9                    LDX     BCPTMR2         ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
ECA1    FF C0 44                    STX     IO+PTMR+T.LATCH2
ECA4    FE BE BB                    LDX     BCPTMR3         ;GET BINARY PRELOAD VALUE FOR TIMER/COUNTER
ECA7    FF C0 46                    STX     IO+PTMR+T.LATCH3
ECAA    39                          RTS                     ;RETURN TO CALLER
                            ;
                            ;
                            ;
                            ;       SUBROUTINE:     INTACIAT        INITIALIZE ACIAT FROM VALUES STORED
                            ;                                       IN SYSTEM RAM
                            ;
ECAB    CE C0 00            INTACIAT LDX    #IO+ACIAT       ;POINT TO ACIA T
ECAE    86 03                       LDAA    #003H
ECB0    A7 00                       STAA    ACIACR,X        ;RESET THE ACIA
ECB2    B6 BE 80                    LDAA    CRACIAT         ;GET CONTROL WORD FOR ACIA T
ECB5    A7 00                       STAA    ACIACR,X
ECB7    A6 01                       LDAA    ACIARX,X        ;THROW AWAY ANY TRASH ON RECEIVE LINE
ECB9    39                          RTS                     ;RETURN TO CALLER
                            ;
                            ;
                            ;
                            ;       SUBROUTINE:     INTACIAX        INITIALIZE ACIAX FROM VALUES STORED
                            ;                                       IN SYSTEM RAM
                            ;
ECBA    CE C0 08            INTACIAX LDX    #IO+ACIAX       ;POINT TO ACIA X
ECBD    86 03                       LDAA    #003H
ECBF    A7 00                       STAA    ACIACR,X        ;RESET THE ACIA
ECC1    B6 BE 81                    LDAA    CRACIAX         ;GET CONTROL WORD FOR ACIA X
ECC4    A7 00                       STAA    ACIACR,X
ECC6    A6 01                       LDAA    ACIARX,X        ;THROW AWAY ANY TRASH ON RECEIVE LINE
ECC8    39                          RTS                     ;RETURN TO CALLER
                            ;
                            ;
                            ;
                            ;       SUBROUTINE:     INTACIAL        INITIALIZE ACIAL FROM VALUES STORED
                            ;                                       IN SYSTEM RAM
```

```
                          ;
ECC9   CE C0 10          INTACIAL LDX    #IO+ACIAL       ;POINT TO ACIA L
ECCC   86 03                      LDAA   #003H
ECCE   A7 00                      STAA   ACIACR,X        ;RESET THE ACIA
ECD0   B6 BE 82                   LDAA   CRACIAL         ;GET CONTROL WORD FOR ACIA L
ECD3   A7 00                      STAA   ACIACR,X
ECD5   A6 01                      LDAA   ACIARX,X        ;THROW AWAY ANY TRASH ON RECEIVE LINE
ECD7   39                         RTS                    ;RETURN TO CALLER
                          ;
                          ;
                          ;
                          ;
ECD8   BD EB 37          INITPIA  JSR    INTPIAAB
ECDB   BD EB 6D                   JSR    INTPIACD
ECDE   39                         RTS
                          ;
                          ;
ECDF   BD EB A3          INITPTM  JSR    INTPTMAB
ECE2   BD EB CF                   JSR    INTPTMCD
ECE5   BD EB FB                   JSR    INTPTMV
ECE8   BD EC 27                   JSR    INTPTMF
ECEB   BD EC 53                   JSR    INTPTML
ECEE   BD EC 7F                   JSR    INTPTMR
ECF1   39                         RTS
                          ;
                          ;
ECF2   BD EC AB          INITACIA JSR    INTACIAT
ECF5   BD EC BA                   JSR    INTACIAX
ECF8   BD EC C9                   JSR    INTACIAL
ECFB   39                         RTS
                          ;
                          ;
                          ;
                          ;       MAIN INITIALIZATION ROUTINE ON POWER UP
                          ;
                          ;
ECFC   0F                INIT     SEI                    ;MASK INTERRUPTS
ECFD   8E BE 7F                   LDS    #BOS            ;INITIALIZE STACK POINTER FOR SYSTEM
ED00   4F                         CLRA
ED01   43                         COMA
ED02   B7 BE EE                   STAA   INTMASK         ;INTERRUPTS NOT ALLOWED UNTIL INITIALIZATION
ED05   7F BE DC                   CLR    POWERUP         ;AND POWER UP COMPLETE
ED08   CE BF CA                   LDX    #ADR            ;SET UP BLOCK MOVE
ED0B   8E BF EF                   LDS    #PREG           ;(DESTINATION) TO INITIALIZE USER REGISTERS
ED0E   AF 00                      STS    0,X             ;AND SYSTEM VECTORS
ED10   8E F1 AB                   LDS    #BLOCK1         ;POINT TO START OF BLOCK DATA
ED13   AF 02                      STS    2,X
ED15   8E F1 BC                   LDS    #BLOCK2         ;END OF BLOCK
ED18   AF 04                      STS    4,X
ED1A   8E BE 7F                   LDS    #BOS            ;RESET STACK POINTER TO SYSTEM STACK
ED1D   BD E6 46                   JSR    MOVE1           ;DO BLOCK MOVE
ED20   CE BF CA                   LDX    #ADR            ;SET UP BLOCK MOVE
```

```
ED23  8E BE 80             LDS   #CRACIAT     ;(DESTINATION) IO INITIALIZATION PARAMETERS
ED26  AF 00                STS   0,X
ED28  8E F1 BC             LDS   #BLOCK2      ;POINT TO START OF BLOCK DATA
ED2B  AF 02                STS   2,X
ED2D  8E F2 4A             LDS   #BLOCK3      ;END OF BLOCK
ED30  AF 04                STS   4,X
ED32  8E BE 7F             LDS   #BOS         ;RESET STACK POINTER TO SYSTEM STACK
ED35  BD E6 46             JSR   MOVE1        ;DO BLOCK MOVE
ED38  8E BE 7F             LDS   #BOS         ;RESET STACK POINTER TO SYSTEM STACK
ED3B  BD EC D8   INIT2     JSR   INITPIA      ;INITIALIZE PIA'S
ED3E  8E BE 7F             LDS   #BOS         ;RESET STACK POINTER TO SYSTEM STACK
ED41  BD EC DF             JSR   INITPTM      ;INITIALIZE PTM'S
ED44  8E BE 7F             LDS   #BOS         ;RESET STACK POINTER TO SYSTEM STACK
ED47  BD EC F2             JSR   INITACIA     ;INITIALIZE ACIA'S
ED4A  8E BE 7F             LDS   #BOS         ;RESET STACK POINTER TO SYSTEM STACK
ED4D  BD EB 0A             JSR   ROMLAYRAM    ;SIZE MEMORY AND OVERLAY ROM TO RAM
ED50  8E BE 7F             LDS   #BOS         ;RESET STACK POINTER TO SYSTEM STACK
ED53  C6 55                LDAB  #055H        ;POWER UP IS COMPLETE
ED55  F7 BE DC             STAB  POWERUP
ED58  7F BE EE   DONEPOWUP CLR   INTMASK      ;NO INTERRUPT MASK REQUIRED NOW SO
ED5B  0E                   CLI                ;CLEAR INTERRUPT MASK
ED5C  7E E1 EF             JMP   START        ;GO TO PROTO PORTION OF OPERATING SYSTEM
                 ;
                 ;
                 ;
                 ;
                 ;
                 ;         SUBROUTINE:     SONPRINT
                 ;
                 ;                ON ENTRY THE A REGISTER HAS THE VALUE TO PRINT AS A
                 ;                NUMBER FROM 0 TO 25.5
                 ;                THE X REGISTER POINTS TO THE ADDRESS WHERE 4 ASCII BYTES
                 ;                WILL BE PLACED AS A DECIMAL REPRESENTATION OF THE VALUE
                 ;                IN THE A REGISTER.      ACTUAL PRINTING IS PERFORMED BY
                 ;                SOME OTHER ROUTINE.
                 ;
                 ;
ED5F  FF BF 7E   SONPRINT  STX   LSXTEMP      ;SAVE POINTER TO PRINT BUFFER
ED62  CE ED AB             LDX   #LOOKSON     ;GET STARTING ADDRESS OF LOOKUP TABLE
ED65  FF BF 7C             STX   LKSONTMP     ;SAVE IT IN A WORKING TEMP REGISTER
ED68  48                   ASLA               ;IF VALUE OF A REGISTER IS ABOVE 127
ED69  25 28                BCS   SPA128       ;ADD 512 TO LOOKUP POINTER ELSE
ED6B  48         SP1       ASLA               ;IF VALUE OF A REGISTER WAS ABOVE 64
ED6C  25 31                BCS   SPA64        ;ADD 256 TO LOOKUP POINTER ELSE
ED6E  5F         SP2       CLRB               ;ADD REMAINING VALUE OF A REGISTER TO
ED6F  BB BF 7D             ADDA  LKSONTMP+1   ;LOOKUP POINTER
ED72  B7 BF 7D             STAA  LKSONTMP+1
ED75  F9 BF 7C             ADCB  LKSONTMP     ;LOOKUP POINTER NOW POINTS 4*A BYTES
ED78  F7 BF 7C             STAB  LKSONTMP     ;INTO THE LOOKUP TABLE
ED7B  C6 04                LDAB  #04H         ;SET UP TO TRANSFER TO PRINT BUFFER
ED7D  FE BF 7C   SP3       LDX   LKSONTMP     ;FOUR ASCII BYTES
ED80  A6 00                LDAA  0,X          ;GET A BYTE
```

```
ED82  08                   INX                     ;POINT TO NEXT BYTE IN TABLE
ED83  FF BF 7C             STX     LKSONTMP        ;SAVE TABLE POINTER
ED86  FE BF 7E             LDX     LSXTEMP         ;GET PRINT BUFFER POINTER
ED89  A7 00                STAA    0,X             ;PUT ASCII CHARACTER INTO PRINT BUFFER
ED8B  08                   INX                     ;POINT TO NEXT POSITION OF PRINT BUFFER
ED8C  FF BF 7E             STX     LSXTEMP         ;SAVE PRINT BUFFER POINTER
ED8F  5A                   DECB                    ;ONE LESS BYTE TO GO
ED90  26 EB                BNE     SP3             ;LOOP BACK IF NOT THRU WITH FOUR BYTES
ED92  39                   RTS                     ;ELSE RETURN TO CALLER
ED93  36          SPA128   PSHA                    ;SAVE VALUE TO BE PRINTED
ED94  86 02                LDAA    #02H            ;ADD 512 TO INDEX POINTING INTO THE PRINT TABLE
ED96  BB BF 7C             ADDA    LKSONTMP
ED99  B7 BF 7C             STAA    LKSONTMP        ;SAVE THE POINTER
ED9C  32                   PULA                    ;RESTORE VALUE TO BE PRINTED
ED9D  20 CC                BRA     SP1             ;CONTINUE FINDING INDEX INTO PRINT TABLE
ED9F  36          SPA64    PSHA                    ;SAVE VALUE TO BE PRINTED
EDA0  86 01                LDAA    #01H            ;ADD 256 TO INDEX POINTING INTO THE PRINT TABLE
EDA2  BB BF 7C             ADDA    LKSONTMP
EDA5  B7 BF 7C             STAA    LKSONTMP
EDA8  32                   PULA                    ;RESTORE VALUE TO BE PRINTED
EDA9  20 C3                BRA     SP2             ;CONTINUE FINDING INDEX INTO PRINT TABLE
                  :
                  :
EDAB  20 30 20 20 LOOKSON  FCC     ' 0  '          ;SONAR HEX TO ASCII DECIMAL LOOKUP TABLE
EDAF  20 30 2E 31          FCC     ' 0.1'
EDB3  20 30 2E 32          FCC     ' 0.2'
EDB7  20 30 2E 33          FCC     ' 0.3'
EDBB  20 30 2E 34          FCC     ' 0.4'
EDBF  20 30 2E 35          FCC     ' 0.5'
EDC3  20 30 2E 36          FCC     ' 0.6'
EDC7  20 30 2E 37          FCC     ' 0.7'
EDCB  20 30 2E 38          FCC     ' 0.8'
EDCF  20 30 2E 39          FCC     ' 0.9'
EDD3  20 31 2E 30          FCC     ' 1.0'
EDD7  20 31 2E 31          FCC     ' 1.1'
EDDB  20 31 2E 32          FCC     ' 1.2'
EDDF  20 31 2E 33          FCC     ' 1.3'
EDE3  20 31 2E 34          FCC     ' 1.4'
EDE7  20 31 2E 35          FCC     ' 1.5'
EDEB  20 31 2E 36          FCC     ' 1.6'
EDEF  20 31 2E 37          FCC     ' 1.7'
EDF3  20 31 2E 38          FCC     ' 1.8'
EDF7  20 31 2E 39          FCC     ' 1.9'
EDFB  20 32 2E 30          FCC     ' 2.0'
EDFF  20 32 2E 31          FCC     ' 2.1'
EE03  20 32 2E 32          FCC     ' 2.2'
EE07  20 32 2E 33          FCC     ' 2.3'
EE0B  20 32 2E 34          FCC     ' 2.4'
EE0F  20 32 2E 35          FCC     ' 2.5'
EE13  20 32 2E 36          FCC     ' 2.6'
EE17  20 32 2E 37          FCC     ' 2.7'
EE1B  20 32 2E 38          FCC     ' 2.8'
```

```
EE1F   20 32 2E 39         FCC    ' 2.9'
EE23   20 33 2E 30         FCC    ' 3.0'
EE27   20 33 2E 31         FCC    ' 3.1'
EE2B   20 33 2E 32         FCC    ' 3.2'
EE2F   20 33 2E 33         FCC    ' 3.3'
EE33   20 33 2E 34         FCC    ' 3.4'
EE37   20 33 2E 35         FCC    ' 3.5'
EE3B   20 33 2E 36         FCC    ' 3.6'
EE3F   20 33 2E 37         FCC    ' 3.7'
EE43   20 33 2E 38         FCC    ' 3.8'
EE47   20 33 2E 39         FCC    ' 3.9'
EE4B   20 34 2E 30         FCC    ' 4.0'
EE4F   20 34 2E 31         FCC    ' 4.1'
EE53   20 34 2E 32         FCC    ' 4.2'
EE57   20 34 2E 33         FCC    ' 4.3'
EE5B   20 34 2E 34         FCC    ' 4.4'
EE5F   20 34 2E 35         FCC    ' 4.5'
EE63   20 34 2E 36         FCC    ' 4.6'
EE67   20 34 2E 37         FCC    ' 4.7'
EE6B   20 34 2E 38         FCC    ' 4.8'
EE6F   20 34 2E 39         FCC    ' 4.9'
EE73   20 35 2E 30         FCC    ' 5.0'
EE77   20 35 2E 31         FCC    ' 5.1'
EE7B   20 35 2E 32         FCC    ' 5.2'
EE7F   20 35 2E 33         FCC    ' 5.3'
EE83   20 35 2E 34         FCC    ' 5.4'
EE87   20 35 2E 35         FCC    ' 5.5'
EE8B   20 35 2E 36         FCC    ' 5.6'
EE8F   20 35 2E 37         FCC    ' 5.7'
EE93   20 35 2E 38         FCC    ' 5.8'
EE97   20 35 2E 39         FCC    ' 5.9'
EE9B   20 36 2E 30         FCC    ' 6.0'
EE9F   20 36 2E 31         FCC    ' 6.1'
EEA3   20 36 2E 32         FCC    ' 6.2'
EEA7   20 36 2E 33         FCC    ' 6.3'
EEAB   20 36 2E 34         FCC    ' 6.4'
EEAF   20 36 2E 35         FCC    ' 6.5'
EEB3   20 36 2E 36         FCC    ' 6.6'
EEB7   20 36 2E 37         FCC    ' 6.7'
EEBB   20 36 2E 38         FCC    ' 6.8'
EEBF   20 36 2E 39         FCC    ' 6.9'
EEC3   20 37 2E 30         FCC    ' 7.0'
EEC7   20 37 2E 31         FCC    ' 7.1'
EECB   20 37 2E 32         FCC    ' 7.2'
EECF   20 37 2E 33         FCC    ' 7.3'
EED3   20 37 2E 34         FCC    ' 7.4'
EED7   20 37 2E 35         FCC    ' 7.5'
EEDB   20 37 2E 36         FCC    ' 7.6'
EEDF   20 37 2E 37         FCC    ' 7.7'
EEE3   20 37 2E 38         FCC    ' 7.8'
EEE7   20 37 2E 39         FCC    ' 7.9'
EEEB   20 38 2E 30         FCC    ' 8.0'
```

```
EEEF   20 38 2E 31        FCC   ' 8.1'
EEF3   20 38 2E 32        FCC   ' 8.2'
EEF7   20 38 2E 33        FCC   ' 8.3'
EEFB   20 38 2E 34        FCC   ' 8.4'
EEFF   20 38 2E 35        FCC   ' 8.5'
EF03   20 38 2E 36        FCC   ' 8.6'
EF07   20 38 2E 37        FCC   ' 8.7'
EF0B   20 38 2E 38        FCC   ' 8.8'
EF0F   20 38 2E 39        FCC   ' 8.9'
EF13   20 39 2E 30        FCC   ' 9.0'
EF17   20 39 2E 31        FCC   ' 9.1'
EF1B   20 39 2E 32        FCC   ' 9.2'
EF1F   20 39 2E 33        FCC   ' 9.3'
EF23   20 39 2E 34        FCC   ' 9.4'
EF27   20 39 2E 35        FCC   ' 9.5'
EF2B   20 39 2E 36        FCC   ' 9.6'
EF2F   20 39 2E 37        FCC   ' 9.7'
EF33   20 39 2E 38        FCC   ' 9.8'
EF37   20 39 2E 39        FCC   ' 9.9'
EF3B   31 30 2E 30        FCC   '10.0'
EF3F   31 30 2E 31        FCC   '10.1'
EF43   31 30 2E 32        FCC   '10.2'
EF47   31 30 2E 33        FCC   '10.3'
EF4B   31 30 2E 34        FCC   '10.4'
EF4F   31 30 2E 35        FCC   '10.5'
EF53   31 30 2E 36        FCC   '10.6'
EF57   31 30 2E 37        FCC   '10.7'
EF5B   31 30 2E 38        FCC   '10.8'
EF5F   31 30 2E 39        FCC   '10.9'
EF63   31 31 2E 30        FCC   '11.0'
EF67   31 31 2E 31        FCC   '11.1'
EF6B   31 31 2E 32        FCC   '11.2'
EF6F   31 31 2E 33        FCC   '11.3'
EF73   31 31 2E 34        FCC   '11.4'
EF77   31 31 2E 35        FCC   '11.5'
EF7B   31 31 2E 36        FCC   '11.6'
EF7F   31 31 2E 37        FCC   '11.7'
EF83   31 31 2E 38        FCC   '11.8'
EF87   31 31 2E 39        FCC   '11.9'
EF8B   31 32 2E 30        FCC   '12.0'
EF8F   31 32 2E 31        FCC   '12.1'
EF93   31 32 2E 32        FCC   '12.2'
EF97   31 32 2E 33        FCC   '12.3'
EF9B   31 32 2E 34        FCC   '12.4'
EF9F   31 32 2E 35        FCC   '12.5'
EFA3   31 32 2E 36        FCC   '12.6'
EFA7   31 32 2E 37        FCC   '12.7'
EFAB   31 32 2E 38        FCC   '12.8'
EFAF   31 32 2E 39        FCC   '12.9'
EFB3   31 33 2E 30        FCC   '13.0'
EFB7   31 33 2E 31        FCC   '13.1'
EFBB   31 33 2E 32        FCC   '13.2'
```

```
EFBF   31 33 2E 33         FCC    '13.3'
EFC3   31 33 2E 34         FCC    '13.4'
EFC7   31 33 2E 35         FCC    '13.5'
EFCB   31 33 2E 36         FCC    '13.6'
EFCF   31 33 2E 37         FCC    '13.7'
EFD3   31 33 2E 38         FCC    '13.8'
EFD7   31 33 2E 39         FCC    '13.9'
EFDB   31 34 2E 30         FCC    '14.0'
EFDF   31 34 2E 31         FCC    '14.1'
EFE3   31 34 2E 32         FCC    '14.2'
EFE7   31 34 2E 33         FCC    '14.3'
EFEB   31 34 2E 34         FCC    '14.4'
EFEF   31 34 2E 35         FCC    '14.5'
EFF3   31 34 2E 36         FCC    '14.6'
EFF7   31 34 2E 37         FCC    '14.7'
EFFB   31 34 2E 38         FCC    '14.8'
EFFF   31 34 2E 39         FCC    '14.9'
F003   31 35 2E 30         FCC    '15.0'
F007   31 35 2E 31         FCC    '15.1'
F00B   31 35 2E 32         FCC    '15.2'
F00F   31 35 2E 33         FCC    '15.3'
F013   31 35 2E 34         FCC    '15.4'
F017   31 35 2E 35         FCC    '15.5'
F01B   31 35 2E 36         FCC    '15.6'
F01F   31 35 2E 37         FCC    '15.7'
F023   31 35 2E 38         FCC    '15.8'
F027   31 35 2E 39         FCC    '15.9'
F02B   31 36 2E 30         FCC    '16.0'
F02F   31 36 2E 31         FCC    '16.1'
F033   31 36 2E 32         FCC    '16.2'
F037   31 36 2E 33         FCC    '16.3'
F03B   31 36 2E 34         FCC    '16.4'
F03F   31 36 2E 35         FCC    '16.5'
F043   31 36 2E 36         FCC    '16.6'
F047   31 36 2E 37         FCC    '16.7'
F04B   31 36 2E 38         FCC    '16.8'
F04F   31 36 2E 39         FCC    '16.9'
F053   31 37 2E 30         FCC    '17.0'
F057   31 37 2E 31         FCC    '17.1'
F05B   31 37 2E 32         FCC    '17.2'
F05F   31 37 2E 33         FCC    '17.3'
F063   31 37 2E 34         FCC    '17.4'
F067   31 37 2E 35         FCC    '17.5'
F06B   31 37 2E 36         FCC    '17.6'
F06F   31 37 2E 37         FCC    '17.7'
F073   31 37 2E 38         FCC    '17.8'
F077   31 37 2E 39         FCC    '17.9'
F07B   31 38 2E 30         FCC    '18.0'
F07F   31 38 2E 31         FCC    '18.1'
F083   31 38 2E 32         FCC    '18.2'
F087   31 38 2E 33         FCC    '18.3'
F08B   31 38 2E 34         FCC    '18.4'
```

```
F08F   31 38 2E 35          FCC   '18.5'
F093   31 38 2E 36          FCC   '18.6'
F097   31 38 2E 37          FCC   '18.7'
F09B   31 38 2E 38          FCC   '18.8'
F09F   31 38 2E 39          FCC   '18.9'
F0A3   31 39 2E 30          FCC   '19.0'
F0A7   31 39 2E 31          FCC   '19.1'
F0AB   31 39 2E 32          FCC   '19.2'
F0AF   31 39 2E 33          FCC   '19.3'
F0B3   31 39 2E 34          FCC   '19.4'
F0B7   31 39 2E 35          FCC   '19.5'
F0BB   31 39 2E 36          FCC   '19.6'
F0BF   31 39 2E 37          FCC   '19.7'
F0C3   31 39 2E 38          FCC   '19.8'
F0C7   31 39 2E 39          FCC   '19.9'
F0CB   32 30 2E 30          FCC   '20.0'
F0CF   32 30 2E 31          FCC   '20.1'
F0D3   32 30 2E 32          FCC   '20.2'
F0D7   32 30 2E 33          FCC   '20.3'
F0DB   32 30 2E 34          FCC   '20.4'
F0DF   32 30 2E 35          FCC   '20.5'
F0E3   32 30 2E 36          FCC   '20.6'
F0E7   32 30 2E 37          FCC   '20.7'
F0EB   32 30 2E 38          FCC   '20.8'
F0EF   32 30 2E 39          FCC   '20.9'
F0F3   32 31 2E 30          FCC   '21.0'
F0F7   32 31 2E 31          FCC   '21.1'
F0FB   32 31 2E 32          FCC   '21.2'
F0FF   32 31 2E 33          FCC   '21.3'
F103   32 31 2E 34          FCC   '21.4'
F107   32 31 2E 35          FCC   '21.5'
F10B   32 31 2E 36          FCC   '21.6'
F10F   32 31 2E 37          FCC   '21.7'
F113   32 31 2E 38          FCC   '21.8'
F117   32 31 2E 39          FCC   '21.9'
F11B   32 32 2E 30          FCC   '22.0'
F11F   32 32 2E 31          FCC   '22.1'
F123   32 32 2E 32          FCC   '22.2'
F127   32 32 2E 33          FCC   '22.3'
F12B   32 32 2E 34          FCC   '22.4'
F12F   32 32 2E 35          FCC   '22.5'
F133   32 32 2E 36          FCC   '22.6'
F137   32 32 2E 37          FCC   '22.7'
F13B   32 32 2E 38          FCC   '22.8'
F13F   32 32 2E 39          FCC   '22.9'
F143   32 33 2E 30          FCC   '23.0'
F147   32 33 2E 31          FCC   '23.1'
F14B   32 33 2E 32          FCC   '23.2'
F14F   32 33 2E 33          FCC   '23.3'
F153   32 33 2E 34          FCC   '23.4'
F157   32 33 2E 35          FCC   '23.5'
F15B   32 33 2E 36          FCC   '23.6'
```

```
F15F   32 33 2E 37          FCC   '23.7'
F163   32 33 2E 38          FCC   '23.8'
F167   32 33 2E 39          FCC   '23.9'
F16B   32 34 2E 30          FCC   '24.0'
F16F   32 34 2E 31          FCC   '24.1'
F173   32 34 2E 32          FCC   '24.2'
F177   32 34 2E 33          FCC   '24.3'
F17B   32 34 2E 34          FCC   '24.4'
F17F   32 34 2E 35          FCC   '24.5'
F183   32 34 2E 36          FCC   '24.6'
F187   32 34 2E 37          FCC   '24.7'
F18B   32 34 2E 38          FCC   '24.8'
F18F   32 34 2E 39          FCC   '24.9'
F193   32 35 2E 30          FCC   '25.0'
F197   32 35 2E 31          FCC   '25.1'
F19B   32 35 2E 32          FCC   '25.2'
F19F   32 35 2E 33          FCC   '25.3'
F1A3   32 35 2E 34          FCC   '25.4'
F1A7   32 35 2E 35          FCC   '25.5'
                     ;
                     ;
                     ;
                     ;
                     ;      BLOCKS FOR INCLUSION WITH MARRS-1 OPERATING SYSTEM ROM
                     ;
                     ;
F1AB   E1 EF         BLOCK1 FDB START
F1AD   BE 00                FDB USRSTAK
F1AF   E2 A8                FDB SWI30
F1B1   E1 FB                FDB ACIAA
F1B3   7E                   FCB 07EH
F1B4   E9 04                FDB TICHAN
F1B6   7E                   FCB 07EH
F1B7   E2 95                FDB SWIHAN
F1B9   7E                   FCB 07EH
F1BA   E1 F8                FDB BREAK
                     ;
                     ;
F1BC   15            BLOCK2 FCB 15H       ;COPY OF CONTROL REGISTER BYTE FOR ACIAT
F1BD   15                   FCB 15H       ;COPY OF CONTROL REGISTER BYTE FOR ACIAX
F1BE   15                   FCB 15H       ;COPY OF CONTROL REGISTER BYTE FOR ACIAL
                     ;
                     ;
F1BF   00                   FCB 0H        ;COPY OF CONTROL REGISTER BYTE FOR PTMAB1
F1C0   00                   FCB 0H        ;COPY OF CONTROL REGISTER BYTE FOR PTMAB2
F1C1   80                   FCB 080H      ;COPY OF CONTROL REGISTER BYTE FOR PTMAB3
F1C2   00                   FCB 0H        ;COPY OF CONTROL REGISTER BYTE FOR PTMCD1
F1C3   00                   FCB 0H        ;COPY OF CONTROL REGISTER BYTE FOR PTMCD2
F1C4   00                   FCB 0H        ;COPY OF CONTROL REGISTER BYTE FOR PTMCD3
F1C5   80                   FCB 080H      ;COPY OF CONTROL REGISTER BYTE FOR PTMV1
F1C6   80                   FCB 080H      ;COPY OF CONTROL REGISTER BYTE FOR PTMV2
F1C7   00                   FCB 0H        ;COPY OF CONTROL REGISTER BYTE FOR PTMV3
```

```
F1C8   40        FCB 040H     ;COPY OF CONTROL REGISTER BYTE FOR PTMF1
F1C9   40        FCB 040H     ;COPY OF CONTROL REGISTER BYTE FOR PTMF2
F1CA   00        FCB 0H       ;COPY OF CONTROL REGISTER BYTE FOR PTMF3
F1CB   40        FCB 040H     ;COPY OF CONTROL REGISTER BYTE FOR PTML1
F1CC   40        FCB 040H     ;COPY OF CONTROL REGISTER BYTE FOR PTML2
F1CD   00        FCB 0H       ;COPY OF CONTROL REGISTER BYTE FOR PTML3
F1CE   40        FCB 040H     ;COPY OF CONTROL REGISTER BYTE FOR PTMR1
F1CF   40        FCB 040H     ;COPY OF CONTROL REGISTER BYTE FOR PTMR2
F1D0   00        FCB 0H       ;COPY OF CONTROL REGISTER BYTE FOR PTMR3
                 ;
                 ;
F1D1   F5        FCB 0F5H     ;COPY OF CONTROL REGISTER BYTE FOR PIAA
F1D2   04        FCB 04H      ;COPY OF CONTROL REGISTER BYTE FOR PIAB
F1D3   04        FCB 04H      ;COPY OF CONTROL REGISTER BYTE FOR PIAC
F1D4   04        FCB 04H      ;COPY OF CONTROL REGISTER BYTE FOR PIAD
                 ;
                 ;
F1D5   FF F6     FDB 0FFF6H   ;BINARY COUNT PRELOAD FOR PTMAB1
F1D7   FF F6     FDB 0FFF6H   ;BINARY COUNT PRELOAD FOR PTMAB2
F1D9   00 24     FDB 00024H   ;BINARY COUNT PRELOAD FOR PTMAB3
F1DB   FF F6     FDB 0FFF6H   ;BINARY COUNT PRELOAD FOR PTMCD1
F1DD   FF F6     FDB 0FFF6H   ;BINARY COUNT PRELOAD FOR PTMCD2
F1DF   FF FF     FDB 0FFFFH   ;BINARY COUNT PRELOAD FOR PTMCD3
F1E1   52 07     FDB 05207H   ;BINARY COUNT PRELOAD FOR PTMV1
F1E3   52 07     FDB 05207H   ;BINARY COUNT PRELOAD FOR PTMV2
F1E5   FF FF     FDB 0FFFFH   ;BINARY COUNT PRELOAD FOR PTMV3
F1E7   00 03     FDB 00003H   ;BINARY COUNT PRELOAD FOR PTMF1
F1E9   00 03     FDB 00003H   ;BINARY COUNT PRELOAD FOR PTMF2
F1EB   FF FF     FDB 0FFFFH   ;BINARY COUNT PRELOAD FOR PTMF3
F1ED   00 3F     FDB 0003FH   ;BINARY COUNT PRELOAD FOR PTML1
F1EF   00 3F     FDB 0003FH   ;BINARY COUNT PRELOAD FOR PTML2
F1F1   FF FF     FDB 0FFFFH   ;BINARY COUNT PRELOAD FOR PTML3
F1F3   00 3F     FDB 0003FH   ;BINARY COUNT PRELOAD FOR PTMR1
F1F5   00 3F     FDB 0003FH   ;BINARY COUNT PRELOAD FOR PTMR2
F1F7   FF FF     FDB 0FFFFH   ;BINARY COUNT PRELOAD FOR PTMR3
                 ;
                 ;
F1F9   01        FCB 01H      ;SONAR SELECT BYTE WRITTEN TO PIA A
F1FA   01        FCB 01H      ;SONAR SELECT BYTE WRITTEN TO PIA B
F1FB   01        FCB 01H      ;SONAR SELECT BYTE WRITTEN TO PIA C
F1FC   01        FCB 01H      ;SONAR SELECT BYTE WRITTEN TO PIA D
                 ;
                 ;
F1FD   12        FCB 012H     ;LAST SONAR A READING
F1FE   12        FCB 012H     ;LAST SONAR B READING
F1FF   12        FCB 012H     ;LAST SONAR C READING
F200   12        FCB 012H     ;LAST SONAR D READING
                 ;
                 ;
F201   00 00     FDB 0000H          ;TIC TIME OF LAST SONAR READING
                 ;
F203   39        FCB 039H           ;JUMP VECTOR FOR EXTENDED TIC INTERRUPT
```

```
                                                      ;ROUTINE (PRESENTLY SET TO RTS)
F204   01 01           FDB 00101H
                  ;
F206   00              FCB 000H                       ;TIME ZERO FROM DRIVE COMPUTER (15 BCD LS NIBBLES)
F207   00              FCB 000H
F208   00              FCB 000H
F209   00              FCB 000H
F20A   02              FCB 002H
F20B   0A              FCB 00AH
F20C   05              FCB 005H
F20D   04              FCB 004H
F20E   01              FCB 001H
F20F   02              FCB 002H
F210   01              FCB 001H
F211   04              FCB 004H
F212   08              FCB 008H
F213   09              FCB 009H
F214   01              FCB 001H
                  ;
F215   00 00           FDB 00000H                     ;ONE TENTH SECOND TIC TIME COUNT SINCE TIME ZERO
                  ;
F217   48              FCB 048H                       ;OFFSET INTO I/O AREA FOR PORT INITIALIZATION
                  ;
F218   00              FCB 000H                       ;POWERUP NOT COMPLETE
                  ;
F219   00 00           FDB 00000H                     ;START OF RAM
F21B   C0 00           FDB IO                         ;END OF RAM + 1 (I/O AREA)
F21D   55              FCB 055H                       ;CHANGE SELECTED SONARS IF 00 ELSE INTERRUPT
                                                      ;HANDLER SETS THIS BYTE TO 055H
F21E   01              FCB 001H                       ;NEXT SONAR SELECT BYTE WRITTEN TO PIA A
F21F   01              FCB 001H                       ;NEXT SONAR SELECT BYTE WRITTEN TO PIA B
F220   01              FCB 001H                       ;NEXT SONAR SELECT BYTE WRITTEN TO PIA C
F221   01              FCB 001H                       ;NEXT SONAR SELECT BYTE WRITTEN TO PIA D
F222   00 00           FDB     0              ;TICTEMP0 TEMPORARY REGISTER FOR TIC INTERRUPT
F224   00 00           FDB     0              ;TICTEMP1 TEMPORARY REGISTER FOR TIC INTERRUPT
F226   00 00           FDB     0              ;TICTEMP2 TEMPORARY REGISTER FOR TIC INTERRUPT
F228   00 00           FDB     0              ;TICTEMP3 TEMPORARY REGISTER FOR TIC INTERRUPT
F22A   FF              FCB     0FFH           ;INTMASK INTERRUPTS MASKED IF $FF ELSE NOT MASKED IF $00
F22B   00              FCB     0              ;SONDATSA SONAR SELECT BYTE FROM READSONAR ROUTINE
F22C   00              FCB     0              ;SONDATRA SONAR READING FROM READSONAR ROUTINE
F22D   00              FCB     0              ;SONDATSB SONAR SELECT BYTE FROM READSONAR ROUTINE
F22E   00              FCB     0              ;SONDATRB SONAR READING FROM READSONAR ROUTINE
F22F   00              FCB     0              ;SONDATSC SONAR SELECT BYTE FROM READSONAR ROUTINE
F230   00              FCB     0              ;SONDATRC SONAR READING FROM READSONAR ROUTINE
F231   00              FCB     0              ;SONDATSD SONAR SELECT BYTE FROM READSONAR ROUTINE
F232   00              FCB     0              ;SONDATRD SONAR READING FROM READSONAR ROUTINE
F233   00 49           FDB     0049H          ;FWCOUNT FRONT WHEEL ABSOLUTE COUNT FOR STRAIGHT AHEAD
F235   00 00           FDB     0              ;RWCOUNT1 RIGHT WHEEL PTM COUNTER 1 COUNTS
F237   00 00           FDB     0              ;RWCOUNT2 RIGHT WHEEL PTM COUNTER 2 COUNTS
F239   00 00           FDB     0              ;LWCOUNT1 LEFT  WHEEL PTM COUNTER 1 COUNTS
F23B   00 00           FDB     0              ;LWCOUNT2 LEFT  WHEEL PTM COUNTER 2 COUNTS
F23D   00              FCB     0              ;RWMOD1  RIGHT WHEEL PTM 1 MODULO COUNT
```

```
F23E  00                FCB   0      ;RWMOD2  RIGHT WHEEL PTM 2 MODULO COUNT
F23F  00                FCB   0      ;LWMOD1  LEFT  WHEEL PTM 1 MODULO COUNT
F240  00                FCB   0      ;LWMOD2  LEFT  WHEEL PTM 2 MODULO COUNT
F241  7E                FCB   07EH
F242  01 00             FDB   00100H ;JUMP VECTOR FOR C (COLD START) COMMAND
F244  7E                FCB   07EH
F245  01 03             FDB   00103H ;JUMP VECTOR FOR W (WARM START) COMMAND
F247  7E                FCB   07EH
F248  20 01             FDB   02001H ;JUMP VECTOR FOR O (ODESSEY) COMMAND
F24A  01          BLOCK3 NOP                    ;MARK END OF BLOCK 2
                  ;
                  ;
                  ;
                  ;
FF00                    ORG 0FF00H              ;MARRS-1 SYSTEM CREDIT AREA
FF00  4D 41 52 52       FCC 'MARRS-1 CREATORS'
FF04  53 20 31 20
FF08  43 52 45 41
FF0C  54 4F 52 53
FF10  31 4C 54 20       FCC '1LT RANDALL OWEN'
FF14  52 41 4E 44
FF18  41 4C 4C 20
FF1C  4F 57 45 4E
FF20  43 41 50 54       FCC 'CAPT            '
FF24  20 20 20 20
FF28  20 20 20 20
FF2C  20 20 20 20
FF30  20 20 42 45       FCC '  BERT          '
FF34  52 54 20 20
FF38  20 20 20 20
FF3C  20 20 20 20
FF40  20 20 20 20       FCC '    SCHNEIDER   '
FF44  53 43 48 4E
FF48  45 49 44 45
FF4C  52 20 20 20
FF50  31 4C 54 20       FCC '1LT TOM CLIFFORD'
FF54  54 4F 4D 20
FF58  43 4C 49 46
FF5C  46 4F 52 44
FFF5                    ORG 0FFF5H              ;RESET VECTOR AREA
FFF5  7E EC FC    LASTCH JMP INIT               ;LAST CHANCE FOR RECOVERY
FFF8  BF F7       SYSIRQ FDB IRQVEC             ; " FOR MASKABLE INTERRUPT REQUEST
FFFA  BF FA       SYSSWI FDB SWIVEC             ; " FOR SOFTWARE INTERRUPT REQUEST
FFFC  BF FD       SYSNMI FDB NMIVEC             ; " FOR NONMASKABLE IRQ
FFFE  EC FC       RESETV FDB INIT               ; " FOR MASTER AND POWER ON RESET
                  ;
                  ;****************************************************************
                  ;
0000                     end                    ;THAT'S ALL FOLKS

No error(s).
```

SYMBOL TABLE FOR FILE NAVROM.A

A.C	0000	A.L	0010	A.RX	0001	A.S	0000
A.T	0000	A.TX	0001	A.X	0008	A0	0001
A1	0002	A2	0004	A3	0008	A4	0010
A5	0020	A6	0040	A7	0080	ABORT	E287
ACIAA	E1FB	ACIACR	0060	ACIAI	BFF5	ACIAL	0010
ACIARX	0001	ACIASTAT	0000	ACIAT	0000	ACIATX	0001
ACIAX	0008	ADDAB	E71C	ADDABX1	E718	ADDH	BFCE
ADDL	BFCC	ADDRS	E635	ADDZ	E731	ADR	BFCA
ALPNUM	E815	ANNOTOK	E831	ANNUM	E829	ANOK	E825
ANRTS	E833	AREG	BFEC	ASCIIL	E7EC	ASCIIR	E7F0
ASCRTS	E7FA	B0	0001	B1	0002	B2	0004
B3	0008	B4	0010	B5	0020	B6	0040
B7	0080	BADSZMEM	EAE4	BADTAP	E4C1	BCPTMAB1	BE99
BCPTMAB2	BE9B	BCPTMAB3	BE9D	BCPTMCD1	BE9F	BCPTMCD2	BEA1
BCPTMCD3	BEA3	BCPTMF1	BEAB	BCPTMF2	BEAD	BCPTMF3	BEAF
BCPTML1	BEB1	BCPTML2	BEB3	BCPTML3	BEB5	BCPTMR1	BEB7
BCPTMR2	BEB9	BCPTMR3	BEBB	BCPTMV1	BEA5	BCPTMV2	BEA7
BCPTMV3	BEA9	BLOCK1	F1AB	BLOCK2	F1BC	BLOCK3	F24A
BOS	BE7F	BREAK	E1F8	BREAK1	E291	BREG	BFEB
BUF	BF80	BUFEND	BFC7	BUFPTR	BFD0	BYTENM1	E376
BYTENUM	E36A	C0	0001	C1	0002	C2	0004
C3	0008	C4	0010	C5	0020	C6	0040
C7	0080	CHEKSM	E301	CHKFWC1	E937	CHKFWC2	E93D
CHKLWC1	E96B	CHKLWC2	E976	CHKRWC1	E94C	CHKRWC2	E957
CKSUM	BFD4	COLD	BF05	CONENDC	F550	CONFND	E85F
CONHB	E84F	CONNOGD	E87E	CONSLP	E86A	CONV4	E329
COUNT	BFD3	CPRET	E34B	CPRMEM	E338	CRACIAL	BE82
CRACIAT	BE80	CRACIAX	BE81	CREG	BFEA	CRPIAA	BE95
CRPIAB	BE96	CRPIAC	BE97	CRPIAD	BE98	CRPTMAB1	BE83
CRPTMAB2	BE84	CRPTMAB3	BE85	CRPTMCD1	BE86	CRPTMCD2	BE87
CRPTMCD3	BE88	CRPTMF1	BE8C	CRPTMF2	BE8D	CRPTMF3	BE8E
CRPTML1	BE8F	CRPTML2	BE90	CRPTML3	BE91	CRPTMR1	BE92
CRPTMR2	BE93	CRPTMR3	BE94	CRPTMV1	BE89	CRPTMV2	BE8A
CRPTMV3	BE8B	CS1	E30E	CTABLE	E882	D0	0001
D1	0002	D2	0004	D3	0008	D4	0010
D5	0020	D6	0040	D7	0080	DCRET	E91D
DCRMEM	E911	DISMEM	E34C	DISMEM1	E354	DL10	E27D
DLOOP	E275	DNESZMEM	EAFF	DONEPOWU	ED58	DSZMEM2	EB09
ECHO	BFE8	ENDOM	E3E1	EOF	E3E2	EOF1	E3E7
FEE	E91E	FIE	E924	FINDS	E4C3	FOE	E92A
FRONTWHE	E933	FS10	E4C4	FSTSZGD	EAF1	FUM	E930
FWCOUNT	BEF7	FWDONE	E946	FWRESET	E9A4	GETRG1	E401
GETRG3	E407	GETRG4	E41B	GETRNG	E3F0	GO	E420
GO10	E42B	INCMEM	E908	INIT	ECFC	INIT2	ED3B
INITACIA	ECF2	INITPIA	ECD8	INITPTM	ECDF	INK	E627
INLOOP1	E398	INLOOP2	E3A8	INRET	E910	INTACIAL	ECC9
INTACIAT	ECAB	INTACIAX	ECBA	INTMASK	BEEE	INTPIAAB	EB37
INTPIACD	EB6D	INTPTMAB	EBA3	INTPTMCD	EBCF	INTPTMF	EC27
INTPTML	EC53	INTPTMR	EC7F	INTPTMV	EBFB	INWAIT	E838

IO	C000	IOOFFSET	BEDB	IRQVEC	BFF7	ISCTRLX	E325
JBAD	E636	JPINCX	E834	LASTCH	FFF5	LDR10	E49B
LEFTWHEE	E967	LHF4	E4AF	LHF9	E4B3	LKSONTMP	BF7C
LOAD	E42E	LOAD1	E463	LOAD2	E433	LOCVV	E664
LOOFST	E43C	LOOKSON	EDAB	LOOPPOS1	EA9F	LSXTEMP	BF7E
LWCOUNT1	BEFD	LWCOUNT2	BEFF	LWDONE	E984	LWMOD1	BF03
LWMOD2	BF04	LWRESET	E9B5	MA	E6E8	MACIA	0000
MACIAI	BFF5	MBADDR	E047	MBEGA	A002	MBYTE	E055
MBYTECT	A00B	MC1	E044	MC2	E132	MCHANGE	E095
MCHASI	E087	MCKSM	A00A	MCONTL2	E0FF	MCONTRL	E0E3
MENDA	A004	MENDAI	A005	MINIHC	E0BE	MINCH	E078
MINEEE	E1AC	MINHEX	E0AA	MINDEK1	E1BA	MINDEK2	E1BB
MINDEK0	E1B1	MIO	E000	MIOS	E1EA	MIOV	A000
MLOAD	E00A	MLOAD11	E02F	MLOAD15	E03B	MLOAD19	E040
MLOAD3	E013	MMCL	E19D	MMCLOFF	E19C	MMCONT	A011
MMTAPEI	E134	MNIO	A006	MONENT	E242	MONITR	E245
MORETIC	BEC7	MOUT2H	E0BF	MOUT2HS	E0CA	MOUT4HS	E0C8
MOUTCH	E075	MOUTEEE	E1D1	MOUTHL	E067	MOUTHR	E06B
MOUTS	E0CC	MOUTS2	E0CE	MOVE	E639	MOVE1	E646
MOVE2	E63E	MPDATA1	E07E	MPDATA2	E07B	MPOWDWN	E005
MPRINT	E11F	MPUN11	E148	MPUN22	E15A	MPUN23	E15C
MPUN32	E17C	MPUNCH	E130	MPY8	E774	MPYLP	E77C
MPYSHFT	E780	MSAV	E1A5	MSFE	E113	MSFEI	E11D
MSGMON	E28A	MSP	A008	MSTACK	A042	MTEMP	A00E
MTW	A00F	MTW1	A010	MXHI	A00C	MXLOW	A00D
MXTEMP	A012	MYFLAG0	BFE5	MYFLAG1	BFE6	NA1	E4F2
NA3	E506	NEX2D	E4CC	NMIVEC	BFFD	NSSLA	BEE2
NSSLB	BEE3	NSSLC	BEE4	NSSLD	BEE5	NXTADR	E4E6
NXTRTS	E4F1	OC10	E51C	OC20	E528	ODESSEY	BF0B
OFFSET	BFC8	OSEMOD	BF01	OSEREG	BEF7	OSRAM	BE00
OUTCH	E50B	OUTCHX	E509	OUTLOOP	E386	P.AB	0048
P.CD	0050	P.CRA	0001	P.CRB	0003	P.CRC	0001
P.CRD	0003	P.DDA	0000	P.DDB	0002	P.DDC	0000
P.DDD	0002	P.PRA	0000	P.PRB	0002	P.PRC	0000
P.PRD	0002	P.SRA	0001	P.SRB	0003	P.SRC	0001
P.SRD	0003	P4HEXS	E5A5	PBADR	E5ED	PC	E6A0
PCRLF	E52A	PERIOD	E3B4	PEXISTS	E5AC	PHEX	E7C3
PHF20	E546	PIAAB	0048	PIACD	0050	PIACRA	0001
PIACRB	0003	PIACRC	0001	PIACRD	0003	PIADDA	0000
PIADDB	0002	PIADDC	0000	PIADDD	0002	PIAPA	0000
PIAPB	0002	PIAPC	0000	PIAPD	0002	PIASTATA	0001
PIASTATB	0003	PIASTATC	0001	PIASTATD	0003	PINCX	E7D1
PINXRTS	E7D8	PMESS	E7FB	PMSRTS	E80A	POS	EA8E
POSDONE	EAB7	POWERUP	BEDC	PR1	E2CA	PR10	E2DA
PR20	E2E4	PREC10	E585	PREG	BFEF	PREGS	E2C5
PRELPTMF	E986	PRELPTML	E99A	PRELPTMR	E990	PRNTASC	E3B6
PS	E675	PSPACE	E5A7	PTMAB	0018	PTMCD	0020
PTMCR1	0000	PTMCR2	0001	PTMCR3	0000	PTMF	0030
PTML	0038	PTMR	0040	PTMSTAT	0001	PTMV	0028
PULXA	E705	PUNBYTE	E59D	PUNCH	E52E	PUNCH1	E533
PUND10	E558	PUND20	E55A	PUNT2	E197	PUTA	E7E1
PUTAX	E7DC	PUTRDY	E7E2	PX2	E5BC	PXISTX	E5AF

Q6SPACE	E8E4	QBADR	E8A3	QCKSMER	E88E	QEOF	E8B5
QQUES	E8B9	QREGNAM	E8EB	QRNGERR	E8AB	QSI	E8D8
QTAPER	E8C7	QTPEOF	E8D0	RAMSIZEH	BEDF	RAMSIZEL	BED0
RDPRE	E464	RDSONDON	EA8C	READAC1	E319	READSONA	EA54
RECTYP	BFD2	RESETV	FFFE	RESTAK	E2F1	RET39	E328
RIGHTWHE	E948	RNGERR	E414	ROMLAYRA	EB0A	ROMOVERR	EB16
ROZR2P2	E334	RS	E6B6	RSRSR	E65A	RSRXIT	E68C
RT10	E25A	RT20	E261	RT90	E267	RUS10	E2F7
RWCOUNT1	BEF9	RWCOUNT2	BEFB	RWDONE	E965	RWMOD1	BF01
RWMOD2	BF02	RWRESET	E9A8	SCARRY	E810	SETM1	E5F3
SETMEM	E5DF	SETOFF	E5C2	SETOUT	E5D7	SETPUL	E5DE
SETUS	E811	SIZELP1	EABF	SIZEMEM	EAB8	SM	E5F4
SM10	E5FD	SM30	E60E	SONARA	BEC1	SONARB	BEC2
SONARC	BEC3	SONARD	BEC4	SONARSLA	BEBD	SONARSLB	BEBE
SONARSLC	BEBF	SONARSLD	BEC0	SONCHANG	BEE1	SONDATRA	BEF0
SONDATRB	BEF2	SONDATRC	BEF4	SONDATRD	BEF6	SONDATSA	BEEF
SONDATSB	BEF1	SONDATSC	BEF3	SONDATSD	BEF5	SONNEXA	BEE2
SONNEXB	BEE3	SONNEXC	BEE4	SONNEXD	BEE5	SONPRINT	ED5F
SONTIME	BEC5	SP1	ED6B	SP2	ED6E	SP3	ED7D
SPA128	ED93	SPA64	ED9F	SREG	BFF1	SSLA	BEBD
SSLB	BEBE	SSLC	BEBF	SSLD	BEC0	STAB	E6C9
START	E1EF	START1	E1FD	STAUXH	E722	SUBABX1	E742
SWI20	E2A3	SWI30	E2A8	SWI40	E2AF	SWI50	E2B2
SWIHAN	E295	SWIVEC	BFFA	SYSFLAG	BFE7	SYSIRQ	FFF8
SYSNMI	FFFC	SYSSWI	FFFA	T.AB	0018	T.C1	0000
T.C2	0001	T.C3	0000	T.CD	0020	T.CNT1	0002
T.CNT2	0004	T.CNT3	0006	T.F	0030	T.L	0038
T.LATCH1	0002	T.LATCH2	0004	T.LATCH3	0006	T.R	0040
T.S	0001	T.V	0028	TA	E72B	TABX1	E6CF
TCOUNT	BFE9	TEMP0	BFD5	TEMP1	BFD7	TEMP2	BFD9
TEMP3	BFDB	TEMP4	BFDD	TEMP5	BFDF	TEMP6	BFE1
TEMP7	BFE3	TESTZ	E727	TICHAN	E904	TICHAN1	E9C2
TICHAN2	E9CD	TICHAN3	EA38	TICTEMP0	BEE6	TICTEMP1	BEE8
TICTEMP2	BEEA	TICTEMP3	BEEC	TICTIME	BED9	TIMEZERO	BECA
TRC	E6AC	TSUB	E751	TUSRAM	BC00	USRSTAK	BE00
USWI	BFF3	W20	E61A	W30	E626	WAITTY	E611
WARM	BF08	XABX1	E6D9	XREG	BFED	XZERO	0000

151

```
                   ;*******************************************
                   ;*                                         *
                   ;*               APPENDIX  B               *
                   ;*                                         *
                   ;*******************************************
                   ;          FILE: XTEND.A
                   ;          EXTENDED INTERRUPT HANDLER AS OF 20 OCT 1984
                   ;
                   ;
                   ;
BC00        TUSRAM      EQU   0BC00H       ;TOP OF USER RAM
ECFC        INIT        EQU   0ECFCH       ;SYSTEM INITIALIZATION ENTRY POINT
0000        XZERO       EQU   00000H       ;LOAD ZERO INTO INDEX REGISTER
C000        IO          EQU   0C000H       ;IO AREA
0000        ACIAT       EQU   0            ;TERMINAL ACIA OFFSET INTO IO AREA
0000        A.S         EQU   0            ;ACIA STATUS OFFSET
0000        A.C         EQU   0            ;ACIA CONTROL OFFSET
0001        A.TX        EQU   1            ;ACIA TRANSMIT REGISTER OFFSET
0001        A.RX        EQU   1            ;ACIA RECEIVE REGISTER OFFSET
BEEE        INTMASK     EQU   0BEEEH       ;INTERRUPT MASK 0=NO $FF=YES
EA54        READSONAR   EQU   0EA54H       ;READ SONAR SUBROUTINE
BEE1        SONCHANGE   EQU   0BEE1H       ;CHANGE SONARS IF=0 ELSE NO=$55
E7FB        INTPTMV     EQU   0E7FBH
0011        XON         EQU   011H
0013        XOFF        EQU   013H
BEC7        MORETIC     EQU   0BEC7H       ;JUMP VECTOR FOR EXTENDED INTERRUPTS
BED9        TICTIME     EQU   0BED9H       ;TIME SINCE TIME ZERO OR LAST RESET
                                           ;IN TENTHS OF SECONDS OR REPETITIONS
                                           ;OF THE SONAR INTERRUPT CLOCK
E7EC        ASCIIL      EQU   0E7ECH       ;CONVERT LEFT NIBBLE OF A BYTE TO ASCII
E7F0        ASCIIR      EQU   0E7F0H       ;CONVERT RIGHT NIBBLE OF BYTE TO ASCII
BEF7        FWCOUNT     EQU   0BEF7H       ;FRONT WHEEL COUNT
BEF9        RWCOUNT1    EQU   0BEF9H       ;RIGHT WHEEL COUNT BACKWARD
BEFB        RWCOUNT2    EQU   0BEFBH       ;RIGHT WHEEL COUNT FORWARD
BEFD        LWCOUNT1    EQU   0BEFDH       ;LEFTWHEEL COUNT BACKWARD
BEFF        LWCOUNT2    EQU   0BEFFH       ;LEFTWHEEL COUNT FORWARD
BEEF        SONDATSA    EQU   0BEEFH       ;AS IN  'WHERE'S THE ___'
BEF0        SONDATRA    EQU   0BEF0H       ;SONAR RANGE FOR A
BEF1        SONDATSB    EQU   0BEF1H       ;SONAR SELECT BYTE FOR B
BEF2        SONDATRB    EQU   0BEF2H       ;SONAR RANGE FOR B
BEF3        SONDATSC    EQU   0BEF3H       ;SONAR SELECT BYTE FOR C
BEF4        SONDATRC    EQU   0BEF4H       ;SONAR RANGE FOR C
BEF5        SONDATSD    EQU   0BEF5H       ;SONAR SELECT BYET FOR D
BEF6        SONDATRD    EQU   0BEF6H       ;SONAR RANGE FOR D
ED5F        SONPRINT    EQU   0ED5FH       ;SUBROUTINE THAT CONVERTS THE SONAR RANGE
                                           ;IN BINARY TO DECIMAL TENTHS OF FEET
BFCA        ADR         EQU   0BFCAH       ;DESTINATION ADDRESS FOR MOVE COMMAND
BFCC        ADDL        EQU   0BFCCH       ;START OF MOVE ADDRESS FOR MOVE COMMAND
BFCE        ADDH        EQU   0BFCEH       ;END OF MOVE ADDRESS FOR MOVE COMMAND
E646        MOVE1       EQU   0E646H       ;SUBROUTINE ADDRESS OF MOVE COMMAND
BEE2        SONNEXA     EQU   0BEE2H       ;NEXT SONAR TO BE SELECTED FOR A
BEE3        SONNEXB     EQU   0BEE3H       ;NEXT SONAR TO BE SELECTED FOR B
```

```
BEE4              SONNEXC    EQU    0BEE4H        ;NEXT SONAR TO BE SELECTED FOR C
BEE5              SONNEXD    EQU    0BEE5H        ;NEXT SONAR TO BE SELECTED FOR D
                  ;
                  ;
0100                         ORG    00100H
0100   0F         MAININIT   SEI                  ;ALLOW NO INTERRUPTS UNTIL
                                                  ;INITIALIZATION IS COMPLETE
0101   8E BC 00              LDS    #TUSRAM       ;INITIALIZE STACK POINTER
0104   86 B5                 LDAA   #0B5H         ;SET UP CONTROL WORD FOR TERMINAL
0106   B7 C0 00              STAA   IO+ACIAT+A.C  ;ACIA TO ALLOW INTERRUPT GENERATION
                                                  ;ON BOTH TRANSMIT AND RECEIVE
0109   CE 01 21              LDX    #XTEND        ;SET UP MORETIC EXTENDED INTERRUPT
010C   FF BE C8              STX    MORETIC+1     ;HANDLER
010F   86 7E                 LDAA   #07EH
0111   B7 BE C7              STAA   MORETIC
0114   CE 00 00              LDX    #XZERO
0117   FF BE D9              STX    TICTIME
011A   BD EB FB              JSR    INTPTMV       ;REINITIALIZE THE TICTIME COUNTER
011D   0E                    CLI                  ;ALLOW INTERRUPTS
                  ;
                  ;
                  ;
                  ;
011E   3E         TWIDDLE    WAI                  ;DO NOTHING.  WAIT FOR INTERRUPTS
011F   20 FD                 BRA    TWIDDLE
                  ;
                  ;
                  ;####################################################################
                  ;          NOTE THAT THE MAIN PROGRAM IS DOING NOTHING AND          *
                  ;          AS SUCH CAN BE MODIFIED TO TAKE ADVANTAGE OF PROCESSOR   *
                  ;          CYCLES NOT USED BY THE INTERRUPT HANDLER                 *
                  ;####################################################################
                  ;
                  ;
                  ;    SUBROUTINE:    XTEND        AN EXTENTION OF THE EXISTING
                  ;                                NAV COMPUTER INTERRUPT HANDLER.
                  ;                                ITS PURPOSE IS TO GATHER TIME
                  ;                                TAGED SONAR AND OPTICAL SHAFT
                  ;                                ENCODER DATA AND SEND IT OUT
                  ;                                THE TERMINAL PORT TO AN EXTERNAL
                  ;                                COMPUTER FOR POST MISSION
                  ;                                PROCESSING OR REAL TIME ANALYSIS
                  ;                                FOR DRIVE COMPUTER COMMAND GENERATION.
                  ;
                  ;
                  ;
0121   7D BE E1   XTEND      TST    SONCHANGE     ;HAVE SONARS BEEN CHANGED SINCE
                                                  ;LAST REQUEST?
0124   27 3B                 BEQ    XTEND2        ;IF NOT THEN CONTINUE INTERRUPT
                                                  ;PROCESSING
0126   7D 05 6A   CANDATA    TST    BUFLEFT       ;ELSE SEE IF MORE THAN 256 BYTES IN
0129   22 03                 BHI    CALLRDSON     ;IN BUFFFER. IF YES CALL READSONAR
```

```
012B  7E 01 61                JMP   XTEND2         ;ELSE CONTINUE INTERRUPT PROCESSING
012E  4F          CALLRDSON   CLRA
012F  43                      COMA                 ;CREATE A LINE OF SONAR AND OSE TIME
0130  B7 BE EE                STAA  INTMASK        ;TAGED DATA
0133  BD EA 54                JSR   READSONAR      ;GET THE SONAR DATA AND TIME
0136  4F                      CLRA
0137  B7 BE EE                STAA  INTMASK
013A  BD 04 2D                JSR   SONSWITCH      ;SET UP NEXT SONARS TO BE USED
013D  BD 02 43                JSR   PUTTIME        ;PUT TIME IN BUFFER
0140  BD 02 62                JSR   PUTFW          ;PUT FRONT WHEEL COUNT IN BUFFER
0143  BD 02 6C                JSR   PUTLW          ;PUT LEFT WHEEL COUNTS IN BUFFER
0146  BD 02 8B                JSR   PUTRW          ;PUT RIGHT WHEEL COUNTS IN BUFFER
0149  BD 02 AA                JSR   PUTA           ;PUT SONAR READINGS IN BUFFER
014C  BD 02 FA                JSR   PUTB
014F  BD 03 4A                JSR   PUTC
0152  BD 03 9A                JSR   PUTD
0155  BD 03 EA                JSR   PUTLINE        ;PUT THE LINE BUFFER INTO THE SPOOLER
0158  BD 04 00                JSR   DECBLEFT       ;DECREMENT THE SPOOLER BUFFER BY 64
015B  BD 04 0B                JSR   ADDLEFTPR      ;ADD 64 TO NUMBER LEFT TO PRINT
015E  BD 04 16                JSR   ADDBUFNX       ;POINT TO NEXT AVAILABLE SPOOLER ADDRESS
0161  CE 00 00    XTEND2      LDX   #XZERO
0164  BC 05 66                CPX   LEFTPR         ;ARE CHARACTERS AVAILABLE TO TRANSMIT?
0167  26 03                   BNE   XTEND3         ;YES GO TRANSMIT
0169  7E 01 9A                JMP   TSTRX          ;NO.  CONTINUE INTERRUPT PROCESSING.
016C  7D 05 65    XTEND3      TST   MEXOFF         ;IS XOFF IN EFFECT?
016F  26 29                   BNE   TSTRX          ;YES.  GO CHECK RECEIVE BUFFER.
0171  B6 C0 00                LDAA  IO+ACIAT+A.S   ;ELSE CHECK FOR TRANSMIT BUFFER EMPTY.
0174  47                      ASRA
0175  47                      ASRA
0176  24 22                   BCC   TSTRX          ;NOT EMPTY SO CHECK RECEIVE BUFFER.
0178  FE 05 68                LDX   NXCHPR         ;YES EMPTY SO POINT TO NEXT CHARACTER
017B  A6 00                   LDAA  0,X            ;GET IT AND TRANSMIT IT OUT.
017D  B7 C0 01                STAA  IO+ACIAT+A.TX
0180  08                      INX                  ;POINT TO NEXT CHARACTER
0181  8C A0 00                CPX   #SPBUFEND+1    ;IS IT THE END OF THE BUFFER?
0184  26 03                   BNE   UPNXCHPR       ;NO.  RESTORE THE POINTER
0186  FE 05 6E                LDX   SPBUFST        ;YES.  POINT TO THE START OF THE BUFFER
0189  FF 05 68    UPNXCHPR    STX   NXCHPR         ;RESTORE THE POINTER
018C  FE 05 66                LDX   LEFTPR         ;ONE LESS TO PRINT
018F  09                      DEX
0190  FF 05 66                STX   LEFTPR
0193  FE 05 6A                LDX   BUFLEFT        ;ONE MORE FOR THE AVAILABLE BUFFER
0196  08                      INX
0197  FF 05 6A                STX   BUFLEFT
019A  B6 C0 00    TSTRX       LDAA  IO+ACIAT+A.S   ;CHECK TO SEE IF CHARACTER RECEIVED
019D  47                      ASRA
019E  24 25                   BCC   TSTRX2         ;NOTHING RECEIVED
01A0  B6 C0 01                LDAA  IO+ACIAT+A.RX  ;YES.  GET RECEIVED CHARACTER.
01A3  81 11                   CMPA  #011H          ;IS IT XON?
01A5  26 05                   BNE   TSTRX3         ;NO.  CHECK FOR OTHER COMMANDS
01A7  7F 05 65                CLR   MEXOFF         ;YES SO CLEAR THE XOFF FLAG
01AA  20 B5                   BRA   XTEND2         ;AND GO TRY TO TRANSMIT A CHARACTER
```

```
01AC  81 13     TSTRX3  CMPA  #013H        ;IS IT XOFF?
01AE  26 07             BNE   TSTRX4       ;NO.  CHECK FOR OTHER COMMANDS
01B0  86 FF             LDAA  #0FFH        ;YES.  GO SET THE XOFF FLAG
01B2  B7 05 65          STAA  MEXOFF
01B5  20 0E             BRA   TSTRX2       ;END RECEIVE FUNCTION
01B7  81 07     TSTRX4  CMPA  #007         ;IS IT CONTROL G?
01B9  26 0D             BNE   TSTRX5       ;NO.  CHECK FOR OTHER COMMANDS
01BB  BD EB FB          JSR   INTPTMV      ;RESET 10 HZ COUNTER
01BE  DE 00             LDX   XZERO        ;YES.  RESET TICTIME TO ZERO
01C0  FF BE D9          STX   TICTIME
01C3  20 00             BRA   TSTRX2
01C5  39        TSTRX2  RTS                ;END OF INTERRUPT PROCESSING
01C6  01                NOP
01C7  01                NOP
01C8  81 31     TSTRX5  CMPA  #031H        ;IS IT 1?
01CA  26 19             BNE   TSTRX6       ;NO.  CHECK FOR OTHER COMMANDS
                                           ;YES.  DO SONARS 1 AT A TIME
01CC  CE 04 B9          LDX   #PTAB1END    ;POINT TO END OF TABLE
01CF  FF 04 48          STX   SONEND+1
01D2  CE 04 5D          LDX   #PTAB1       ;POINT TO START OF TABLE
01D5  FF 04 4D          STX   SONEND2+1
01D8  FF 04 5B          STX   PTABNEXT
01DB  BD EB FB          JSR   INTPTMV      ;RESET 10 HZ COUNTER
01DE  DE 00             LDX   XZERO        ;YES.  RESET TICTIME TO ZERO
01E0  FF BE D9          STX   TICTIME
01E3  20 E0             BRA   TSTRX2
01E5  81 32     TSTRX6  CMPA  #032H        ;IS IT 2?
01E7  26 19             BNE   TSTRX7       ;NO.  CHECK FOR OTHER COMMANDS
                                           ;YES.  DO SONARS 2 AT A TIME
01E9  CE 04 E9          LDX   #PTAB2END    ;POINT TO END OF TABLE
01EC  FF 04 48          STX   SONEND+1
01EF  CE 04 BD          LDX   #PTAB2       ;POINT TO START OF TABLE
01F2  FF 04 4D          STX   SONEND2+1
01F5  FF 04 5B          STX   PTABNEXT
01F8  BD EB FB          JSR   INTPTMV      ;RESET 10 HZ COUNTER
01FB  DE 00             LDX   XZERO        ;YES.  RESET TICTIME TO ZERO
01FD  FF BE D9          STX   TICTIME
0200  20 C3             BRA   TSTRX2
0202  81 33     TSTRX7  CMPA  #033H        ;IS IT 3?
0204  26 19             BNE   TSTRX8       ;NO.  CHECK FOR OTHER COMMANDS
                                           ;YES.  DO SONARS 3 AT A TIME
0206  CE 05 09          LDX   #PTAB3END    ;POINT TO END OF TABLE
0209  FF 04 48          STX   SONEND+1
020C  CE 04 ED          LDX   #PTAB3       ;POINT TO START OF TABLE
020F  FF 04 4D          STX   SONEND2+1
0212  FF 04 5B          STX   PTABNEXT
0215  BD EB FB          JSR   INTPTMV      ;RESET 10 HZ COUNTER
0218  DE 00             LDX   XZERO        ;YES.  RESET TICTIME TO ZERO
021A  FF BE D9          STX   TICTIME
021D  20 A6             BRA   TSTRX2
021F  81 34     TSTRX8  CMPA  #034H        ;IS IT 4?
0221  26 19             BNE   TSTRX9       ;NO.  CHECK FOR OTHER COMMANDS
```

```
                                                ;YES.  DO SONARS 4 AT A TIME
0223  CE 05 21                 LDX    #PTAB4END ;POINT TO END OF TABLE
0226  FF 04 48                 STX    SONEND+1
0229  CE 05 0D                 LDX    #PTAB4    ;POINT TO START OF TABLE
022C  FF 04 4D                 STX    SONEND2+1
022F  FF 04 5B                 STX    PTABNEXT
0232  BD EB FB                 JSR    INTPTMV   ;RESET 10 HZ COUNTER
0235  DE 00                    LDX    XZERO     ;YES.  RESET TICTIME TO ZERO
0237  FF BE D9                 STX    TICTIME
023A  20 89                    BRA    TSTRX2
023C  81 03        TSTRX9      CMPA   #003H     ;IS IT CONTROL C?
023E  26 85                    BNE    TSTRX2    ;NO.  END RECEIVE FUNCTION
0240  7E EC FC                 JMP    INIT      ;JUMP TO RESET SYSTEM
                   ;
                   ;
                   ;
                   ;      SUBROUTINE:   PUTTIME     PUTS TIME INTO THE LINE BUFFER
0243  CE 05 27     PUTTIME     LDX    #LNTIME
0246  B6 BE D9                 LDAA   TICTIME
0249  BD 02 53                 JSR    PUT2HEX
024C  B6 BE DA                 LDAA   TICTIME+1
024F  BD 02 53                 JSR    PUT2HEX
0252  39                       RTS
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;      SUBROUTINE:   PUT2HEX     CONVERTS A BYTE INTO TWO ASCII
                   ;                                CHARACTERS. ON ENTRY, THE INDEX
                   ;                                REGISTER POINTS TO WHERE THE TWO
                   ;                                ASCII CHARACTERS ARE TO BE PLACED.
0253  36           PUT2HEX     PSHA
0254  BD E7 EC                 JSR    ASCIIL    ;CONVERT THE LEFT NIBBLE
0257  A7 00                    STAA   0,X
0259  08                       INX
025A  32                       PULA
025B  BD E7 F0                 JSR    ASCIIR    ;CONVERT THE RIGHT NIBBLE
025E  A7 00                    STAA   0,X
0260  08                       INX
0261  39                       RTS
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;      SUBROUTINE:   PUTFW       PUT FRONT WHEEL COUNT IN LINE BUFFER
0262  CE 05 2C     PUTFW       LDX    #LNFW
0265  B6 BE F8                 LDAA   FWCOUNT+1
0268  BD 02 53                 JSR    PUT2HEX
026B  39                       RTS
                   ;
```

```
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;      SUBROUTINE:   PUTLW        PUT LEFT WHEEL COUNTS IN LINE BUFFER
026C   CE 05 2F         PUTLW           LDX       #LNBLW1
026F   B6 BE FD                         LDAA      LWCOUNT1         ;GET THE REVERSE COUNTS
0272   BD 02 53                         JSR       PUT2HEX
0275   B6 BE FE                         LDAA      LWCOUNT1+1
0278   BD 02 53                         JSR       PUT2HEX
027B   CE 05 34                         LDX       #LNFLW2          ;GET THE FORWARD COUNTS
027E   B6 BE FF                         LDAA      LWCOUNT2
0281   BD 02 53                         JSR       PUT2HEX
0284   B6 BF 00                         LDAA      LWCOUNT2+1
0287   BD 02 53                         JSR       PUT2HEX
028A   39                               RTS
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;      SUBROUTINE:   PUTRW        PUT RIGHT WHEEL COUNTS IN LINE BUFFER
028B   CE 05 39         PUTRW           LDX       #LNBRW1
028E   B6 BE F9                         LDAA      RWCOUNT1         ;GET REVERSE COUNTS
0291   BD 02 53                         JSR       PUT2HEX
0294   B6 BE FA                         LDAA      RWCOUNT1+1
0297   BD 02 53                         JSR       PUT2HEX
029A   CE 05 3E                         LDX       #LNFRW2          ;GET FORWARD COUNTS
029D   B6 BE FB                         LDAA      RWCOUNT2
02A0   BD 02 53                         JSR       PUT2HEX
02A3   B6 BE FC                         LDAA      RWCOUNT2+1
02A6   BD 02 53                         JSR       PUT2HEX
02A9   39                               RTS
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;      SUBROUTINE:   PUTA         PUT SONAR A SELECTED IN LINE BUFFER
02AA   CE 05 44         PUTA            LDX       #LNATRANS        ;POINT TO LINE POSITION
02AD   B6 BE EF                         LDAA      SONDATSA         ;GET THE SELECTED TRANSDUCER #
```

```
02B0  81 01                 CMPA   #1            ;WAS IT 0
02B2  26 06                 BNE    PUTA2         ;NO CONTINUE SEARCH
02B4  86 30                 LDAA   #'0'
02B6  A7 00                 STAA   0,X
02B8  20 36                 BRA    PUTA8         ;GO PUT DISTANCE READING
02BA  81 02       PUTA2     CMPA   #2            ;WAS IT 1
02BC  26 06                 BNE    PUTA3         ;NO CONTINUE SEARCH
02BE  86 31                 LDAA   #'1'
02C0  A7 00                 STAA   0,X
02C2  20 2C                 BRA    PUTA8         ;GO PUT DISTANCE READING
02C4  81 04       PUTA3     CMPA   #4            ;WAS IT 2
02C6  26 06                 BNE    PUTA4         ;NO CONTINUE SEARCH
02C8  86 32                 LDAA   #'2'
02CA  A7 00                 STAA   0,X
02CC  20 22                 BRA    PUTA8         ;GO PUT DISTANCE READING
02CE  81 10       PUTA4     CMPA   #010H         ;WAS IT 4
02D0  26 06                 BNE    PUTA5         ;NO CONTINUE SEARCH
02D2  86 34                 LDAA   #'4'
02D4  A7 00                 STAA   0,X
02D6  20 18                 BRA    PUTA8         ;GO PUT DISTANCE READING
02D8  81 20       PUTA5     CMPA   #020H         ;WAS IT 5
02DA  26 06                 BNE    PUTA6         ;NO CONTINUE SEARCH
02DC  86 35                 LDAA   #'5'
02DE  A7 00                 STAA   0,X
02E0  20 0E                 BRA    PUTA8         ;GO PUT DISTANCE READING
02E2  81 40       PUTA6     CMPA   #040H         ;WAS IT 6
02E4  26 06                 BNE    PUTA7         ;NO CONTINUE SEARCH
02E6  86 36                 LDAA   #'6'
02E8  A7 00                 STAA   0,X
02EA  20 04                 BRA    PUTA8         ;GO PUT DISTANCE READING
02EC  86 2A       PUTA7     LDAA   #'*'
02EE  A7 00                 STAA   0,X
02F0  B6 BE F0    PUTA8     LDAA   SONDATRA      ;GET THE DISTANCE READING
02F3  CE 05 45              LDX    #LNAREADING   ;POINT TO THE LINE POSITION
02F6  BD ED 5F              JSR    SONPRINT      ;CONVERT THE READING TO DECIMAL
                                                 ;AND PUT IT IN THE LINE BUFFER
02F9  39                    RTS                  ;RETURN TO CALLER
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;    SUBROUTINE:    PUTB       PUT SONAR B SELECTED IN LINE BUFFER
02FA  CE 05 4B    PUTB      LDX    #LNBTRANS     ;POINT TO LINE POSITION
02FD  B6 BE F1              LDAA   SONDATSB      ;GET THE SELECTED TRANSDUCER #
0300  81 01                 CMPA   #1            ;WAS IT 0
0302  26 06                 BNE    PUTB2         ;NO CONTINUE SEARCH
0304  86 30                 LDAA   #'0'
0306  A7 00                 STAA   0,X
0308  20 36                 BRA    PUTB3         ;GO PUT DISTANCE READING
```

```
030A  81 02        PUTB2      CMPA   #2          ;WAS IT 1
030C  26 06                   BNE    PUTB3       ;NO CONTINUE SEARCH
030E  86 31                   LDAA   #'1'
0310  A7 00                   STAA   0,X
0312  20 2C                   BRA    PUTB8       ;GO PUT DISTANCE READING
0314  81 04        PUTB3      CMPA   #4          ;WAS IT 2
0316  26 06                   BNE    PUTB4       ;NO CONTINUE SEARCH
0318  86 32                   LDAA   #'2'
031A  A7 00                   STAA   0,X
031C  20 22                   BRA    PUTB8       ;GO PUT DISTANCE READING
031E  81 10        PUTB4      CMPA   #010H       ;WAS IT 4
0320  26 06                   BNE    PUTB5       ;NO CONTINUE SEARCH
0322  86 34                   LDAA   #'4'
0324  A7 00                   STAA   0,X
0326  20 18                   BRA    PUTB8       ;GO PUT DISTANCE READING
0328  81 20        PUTB5      CMPA   #020H       ;WAS IT 5
032A  26 06                   BNE    PUTB6       ;NO CONTINUE SEARCH
032C  86 35                   LDAA   #'5'
032E  A7 00                   STAA   0,X
0330  20 0E                   BRA    PUTB8       ;GO PUT DISTANCE READING
0332  81 40        PUTB6      CMPA   #040H       ;WAS IT 6
0334  26 06                   BNE    PUTB7       ;NO CONTINUE SEARCH
0336  86 36                   LDAA   #'6'
0338  A7 00                   STAA   0,X
033A  20 04                   BRA    PUTB8       ;GO PUT DISTANCE READING
033C  86 2A        PUTB7      LDAA   #'*'
033E  A7 00                   STAA   0,X
0340  B6 BE F2     PUTB8      LDAA   SONDATRB    ;GET THE DISTANCE READING
0343  CE 05 4C                LDX    #LNBREADING ;POINT TO THE LINE POSITION
0346  BD ED 5F                JSR    SONPRINT    ;CONVERT THE READING TO DECIMAL
                                                 ;AND PUT IT IN THE LINE BUFFER
0349  39                      RTS                ;RETURN TO CALLER
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;    SUBROUTINE:   PUTC       PUT SONAR C SELECTED IN LINE BUFFER
034A  CE 05 52     PUTC       LDX    #LNCTRANS   ;POINT TO LINE POSITION
034D  B6 BE F3                LDAA   SONDATSC    ;GET THE SELECTED TRANSDUCER #
0350  81 01                   CMPA   #1          ;WAS IT 0
0352  26 06                   BNE    PUTC2       ;NO CONTINUE SEARCH
0354  86 30                   LDAA   #'0'
0356  A7 00                   STAA   0,X
0358  20 36                   BRA    PUTC8       ;GO PUT DISTANCE READING
035A  81 02        PUTC2      CMPA   #2          ;WAS IT 1
035C  26 06                   BNE    PUTC3       ;NO CONTINUE SEARCH
035E  86 31                   LDAA   #'1'
0360  A7 00                   STAA   0,X
0362  20 2C                   BRA    PUTC8       ;GO PUT DISTANCE READING
0364  81 04        PUTC3      CMPA   #4          ;WAS IT 2
```

```
0366  26 06                  BNE    PUTC4          ;NO CONTINUE SEARCH
0368  86 32                  LDAA   #'2'
036A  A7 00                  STAA   0,X
036C  20 22                  BRA    PUTC8          ;GO PUT DISTANCE READING
036E  81 10       PUTC4      CMPA   #010H          ;WAS IT 4
0370  26 06                  BNE    PUTC5          ;NO CONTINUE SEARCH
0372  86 34                  LDAA   #'4'
0374  A7 00                  STAA   0,X
0376  20 18                  BRA    PUTC8          ;GO PUT DISTANCE READING
0378  81 20       PUTC5      CMPA   #020H          ;WAS IT 5
037A  26 06                  BNE    PUTC6          ;NO CONTINUE SEARCH
037C  86 35                  LDAA   #'5'
037E  A7 00                  STAA   0,X
0380  20 0E                  BRA    PUTC8          ;GO PUT DISTANCE READING
0382  81 40       PUTC6      CMPA   #040H          ;WAS IT 6
0384  26 06                  BNE    PUTC7          ;NO CONTINUE SEARCH
0386  86 36                  LDAA   #'6'
0388  A7 00                  STAA   0,X
038A  20 04                  BRA    PUTC8          ;GO PUT DISTANCE READING
038C  86 2A       PUTC7      LDAA   #'*'
038E  A7 00                  STAA   0,X
0390  B6 BE F4    PUTC8      LDAA   SONDATRC       ;GET THE DISTANCE READING
0393  CE 05 53               LDX    #LNCREADING    ;POINT TO THE LINE POSITION
0396  BD ED 5F               JSR    SONPRINT       ;CONVERT THE READING TO DECIMAL
                                                   ;AND PUT IT IN THE LINE BUFFER
0399  39                     RTS                   ;RETURN TO CALLER
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;     SUBROUTINE:   PUTD         PUT SONAR D SELECTED IN LINE BUFFER
039A  CE 05 59    PUTD       LDX    #LNDTRANS      ;POINT TO LINE POSITION
039D  B6 BE F5               LDAA   SONDATSD       ;GET THE SELECTED TRANSDUCER #
03A0  81 01                  CMPA   #1             ;WAS IT 0
03A2  26 06                  BNE    PUTD2          ;NO CONTINUE SEARCH
03A4  86 30                  LDAA   #'0'
03A6  A7 00                  STAA   0,X
03A8  20 36                  BRA    PUTD8          ;GO PUT DISTANCE READING
03AA  81 02       PUTD2      CMPA   #2             ;WAS IT 1
03AC  26 06                  BNE    PUTD3          ;NO CONTINUE SEARCH
03AE  86 31                  LDAA   #'1'
03B0  A7 00                  STAA   0,X
03B2  20 2C                  BRA    PUTD8          ;GO PUT DISTANCE READING
03B4  81 04       PUTD3      CMPA   #4             ;WAS IT 2
03B6  26 06                  BNE    PUTD4          ;NO CONTINUE SEARCH
03B8  86 32                  LDAA   #'2'
03BA  A7 00                  STAA   0,X
03BC  20 22                  BRA    PUTD8          ;GO PUT DISTANCE READING
03BE  81 10       PUTD4      CMPA   #010H          ;WAS IT 4
03C0  26 06                  BNE    PUTD5          ;NO CONTINUE SEARCH
```

```
03C2  86 34                    LDAA    #'4'
03C4  A7 00                    STAA    0,X
03C6  20 18                    BRA     PUTD8        ;GO PUT DISTANCE READING
03C8  81 20      PUTD5         CMPA    #020H        ;WAS IT 5
03CA  26 06                    BNE     PUTD6        ;NO CONTINUE SEARCH
03CC  86 35                    LDAA    #'5'
03CE  A7 00                    STAA    0,X
03D0  20 0E                    BRA     PUTD8        ;GO PUT DISTANCE READING
03D2  81 40      PUTD6         CMPA    #040H        ;WAS IT 6
03D4  26 06                    BNE     PUTD7        ;NO CONTINUE SEARCH
03D6  86 36                    LDAA    #'6'
03D8  A7 00                    STAA    0,X
03DA  20 04                    BRA     PUTD8        ;GO PUT DISTANCE READING
03DC  86 2A      PUTD7         LDAA    #'*'
03DE  A7 00                    STAA    0,X
03E0  B6 BE F6   PUTD8         LDAA    SONDATRD     ;GET THE DISTANCE READING
03E3  CE 05 5A                 LDX     #LNDREADING  ;POINT TO THE LINE POSITION
03E6  BD ED 5F                 JSR     SONPRINT     ;CONVERT THE READING TO DECIMAL
                                                    ;AND PUT IT IN THE LINE BUFFER
03E9  39                       RTS                  ;RETURN TO CALLER
                 ;
                 ;
                 ;     SUBROUTINE:  PUTLINE         PUT LINE BUFFER IN SPOOLER BUFFER
03EA  CE 05 25   PUTLINE       LDX     #LNSTART     ;POINT TO START OF LINE BUFFER
03ED  FF BF CC                 STX     ADDL
03F0  CE 05 65                 LDX     #LNEND+6     ;POINT TO END OF LINE BUFFER
03F3  FF BF CE                 STX     ADDH
03F6  FE 05 6C                 LDX     SPBUFNX      ;POINT TO POSITION OF SPOOLER BUFFER
03F9  FF BF CA                 STX     ADR
03FC  BD E6 46                 JSR     MOVE1        ;MOVE THE LINE
03FF  39                       RTS
                 ;
                 ;
                 ;
                 ;
                 ;     SUBROUTINE:  DECBLEFT        DECREMENT BY 64  THE AMOUNT LEFT IN
                 ;                                  THE SPOOLER BUFFER
                 ;
0400  FE 05 6A   DECBLEFT      LDX     BUFLEFT
0403  C6 40                    LDAB    #64
0405  3F                       SWI
0406  0E                       FCB     00EH
0407  FF 05 6A                 STX     BUFLEFT
040A  39                       RTS
                 ;
                 ;
                 ;
                 ;
                 ;
                 ;
                 ;     SUBROUTINE:  ADDLEFTPR       ADD 64 TO THE NUMBER LEFT TO PRINT
                 ;
```

```
                          ;
                          ;
040B  FE 05 66    ADDLEFTPR     LDX    LEFTPR
040E  C6 40                     LDAB   #64
0410  3F                        SWI
0411  0A                        FCB    00AH
0412  FF 05 66                  STX    LEFTPR
0415  39                        RTS
                          ;
                          ;
                          ;
                          ;
                          ;
                          ;     SUBROUTINE:    ADDBUFNX       ADD 64 TO THE NEXT BUFFER POINTER
                          ;
                          ;
0416  FE 05 6C    ADDBUFNX      LDX    SPBUFNX        ;GET THE PRESENT 'NEXT LINE'
                                                      ;POINTER
0419  C6 40                     LDAB   #040H          ;ELSE ADD 64 TO THE POINTER
041B  3F                        SWI
041C  0A                        FCB    00AH
041D  8C A0 00                  CPX    #SPBUFEND+1    ;IS IT THE END OF THE BUFFER?
0420  27 04                     BEQ    ADDBUF2        ;IF YES RESET TO START OF BUFFER
0422  FF 05 6C                  STX    SPBUFNX        ;STORE THE NEW POINTER
0425  39                        RTS                   ;RETURN TO CALLER
0426  CE 20 00    ADDBUF2       LDX    #SPBUFSTAR     ;POINT TO THE START OF THE BUFFER
0429  FF 05 6C                  STX    SPBUFNX        ;STORE THE NEW POINTER
042C  39                        RTS                   ;RETURN TO CALLER
                          ;
                          ;
                          ;
                          ;
                          ;
                          ;     SUBROUTINE:    SONSWITCH      SWITCHES THE SELECTED SONAR
                          ;                         TRANSDUCERS IN A PATTERN CONTAINED IN A
                          ;                         USER DEFINED TABLE.  BEFORE ENTRY, SONCHANGE
                          ;                         MUST BE $55
                          ;
042D  FE 04 5B    SONSWITCH     LDX    PTABNEXT       ;GET POINTER TO NEXT TABLE ENTRY
0430  A6 00                     LDAA   0,X            ;GET TABLE ENTRY FOR A SONAR
0432  B7 BE E2                  STAA   SONNEXA        ;STORE THE TRANSDUCER SELECT BYTE
0435  A6 01                     LDAA   1,X            ;GET TABLE ENTRY FOR B SONAR
0437  B7 BE E3                  STAA   SONNEXB        ;STORE THE TRANSDUCER SELECT BYTE
043A  A6 02                     LDAA   2,X            ;GET TABLE ENTRY FOR C SONAR
043C  B7 BE E4                  STAA   SONNEXC        ;STORE THE TRANSDUCER SELECT BYTE
043F  A6 03                     LDAA   3,X            ;GET TABLE ENTRY FOR D SONAR
0441  B7 BE E5                  STAA   SONNEXD        ;STORE THE TRANSDUCER SELECT BYTE
0444  7F BE E1                  CLR    SONCHANGE      ;TELL TICHAN2 TO CHANGE THE SONARS
0447  8C 04 E9    SONEND        CPX    #PTAB2END      ;IS THIS THE LAST OF THE TABLE?
044A  26 07                     BNE    SONSW2         ;IF NO THEN BRANCH
044C  CE 04 BD    SONEND2       LDX    #PTAB2         ;ELSE RESET TO START OF TABLE
044F  FF 04 5B                  STX    PTABNEXT       ;STORE THE POINTER
```

```
0452  39                            RTS                     ;AND RETURN TO CALLER
0453  08           SONSW2           INX                     ;POINT TO NEXT TABLE ENTRY
0454  08                            INX
0455  08                            INX
0456  08                            INX
0457  FF 04 5B                      STX     PTABNEXT        ;STORE THE POINTER
045A  39                            RTS                     ;RETURN TO CALLER
045B  04 BD        PTABNEXT         FDB     PTAB2
045D  01 00 00 00  PTAB1            FCB     1,0,0,0
0461  00 01 00 00                   FCB     0,1,0,0
0465  00 00 01 00                   FCB     0,0,1,0
0469  00 00 00 01                   FCB     0,0,0,1
046D  02 00 00 00                   FCB     2,0,0,0
0471  00 02 00 00                   FCB     0,2,0,0
0475  00 00 02 00                   FCB     0,0,2,0
0479  00 00 00 02                   FCB     0,0,0,2
047D  04 00 00 00                   FCB     4,0,0,0
0481  00 04 00 00                   FCB     0,4,0,0
0485  00 00 04 00                   FCB     0,0,4,0
0489  00 00 00 04                   FCB     0,0,0,4
048D  10 00 00 00                   FCB     16,0,0,0
0491  00 10 00 00                   FCB     0,16,0,0
0495  00 00 10 00                   FCB     0,0,16,0
0499  00 00 00 10                   FCB     0,0,0,16
049D  20 00 00 00                   FCB     32,0,0,0
04A1  00 20 00 00                   FCB     0,32,0,0
04A5  00 00 20 00                   FCB     0,0,32,0
04A9  00 00 00 20                   FCB     0,0,0,32
04AD  40 00 00 00                   FCB     64,0,0,0
04B1  00 40 00 00                   FCB     0,64,0,0
04B5  00 00 40 00                   FCB     0,0,64,0
04B9  00 00 00 40  PTAB1END         FCB     0,0,0,64
04BD  01 00 01 00  PTAB2            FCB     1,0,1,0
04C1  00 01 00 01                   FCB     0,1,0,1
04C5  02 00 02 00                   FCB     2,0,2,0
04C9  00 02 00 02                   FCB     0,2,0,2
04CD  04 00 04 00                   FCB     4,0,4,0
04D1  00 04 00 04                   FCB     0,4,0,4
04D5  10 00 10 00                   FCB     16,0,16,0
04D9  00 10 00 10                   FCB     0,16,0,16
04DD  20 00 20 00                   FCB     32,0,32,0
04E1  00 20 00 20                   FCB     0,32,0,32
04E5  40 00 40 00                   FCB     64,0,64,0
04E9  00 40 00 40  PTAB2END         FCB     0,64,0,64
04ED  01 02 04 00  PTAB3            FCB     1,2,4,0
04F1  02 04 00 01                   FCB     2,4,0,1
04F5  04 00 01 02                   FCB     4,0,1,2
04F9  00 01 02 04                   FCB     0,1,2,4
04FD  10 20 40 00                   FCB     16,32,64,0
0501  20 40 00 10                   FCB     32,64,0,16
0505  40 00 10 20                   FCB     64,0,16,32
0509  00 10 20 40  PTAB3END         FCB     0,16,32,64
```

```
050D  01 01 01 01   PTAB4        FCB   1,1,1,1
0511  02 02 02 02                FCB   2,2,2,2
0515  04 04 04 04                FCB   4,4,4,4
0519  10 10 10 10                FCB   16,16,16,16
051D  20 20 20 20                FCB   32,32,32,32
0521  40 40 40 40   PTAB4END     FCB   64,64,64,64
                    ;      THE FOLLOWING FORMAT IS USED FOR THE DATA FILE:
                    ;      /time/fw/lw1/lw2/rw1/rw2/A#___/B#___/C#___/D#___/ <CR> <LF>
                    ;      WHERE time    = TENTH'S OF SECONDS COUNT - 2 BYTES
                    ;            fw      = FRONT WHEEL POSITION    - 1 BYTE
                    ;            lw#,rw# = REAR WHEEL COUNTS        - 2 BYTES EA.
                    ;            A#___   = SONAR NUMBER AND READING - 4 BYTES EA.
                    ;
                    ;
0525  09            LNSTART      FCB   09H
0526  2F            LNBUF        FCC   '/'
0527  74 69 6D 65   LNTIME       FCC   'time/'
052B  2F
052C  66 77 2F      LNFW         FCC   'fw/'
052F  62 6C 77 31   LNBLW1       FCC   'blw1/'
0533  2F
0534  66 6C 77 32   LNFLW2       FCC   'flw2/'
0538  2F
0539  62 72 77 31   LNBRW1       FCC   'brw1/'
053D  2F
053E  66 72 77 32   LNFRW2       FCC   'frw2/A'
0542  2F 41
0544  23            LNATRANS     FCC   '#'
0545  5F 5F 5F 5F   LNAREADING   FCC   '____/B'
0549  2F 42
054B  23            LNBTRANS     FCC   '#'
054C  5F 5F 5F 5F   LNBREADING   FCC   '____/C'
0550  2F 43
0552  23            LNCTRANS     FCC   '#'
0553  5F 5F 5F 5F   LNCREADING   FCC   '____/D'
0557  2F 44
0559  23            LNCTRANS     FCC   '#'
055A  5F 5F 5F 5F   LNDREADING   FCC   '____/'
055E  2F
055F  0D 0A 00 00   LNEND        FCB   0DH,0AH,0,0,0,0
0563  00 00
0565  00            MEXOFF       FCB   0
0566  00 00         LEFTPR       FDB   0
0568  20 00         NXCHPR       FDB   SPBUFSTAR
056A  80 00         BUFLEFT      FDB   08000H
056C  20 00         SPBUFNX      FDB   SPBUFSTAR
056E  20 00         SPBUFST      FDB   SPBUFSTAR
2000                             ORG   02000H
2000                SPBUFSTAR    RMB   07FFFH
9FFF                SPBUFEND     END
```

No error(s).

SYMBOL TABLE FOR FILE XTEND.A

A.C	0000	A.RX	0001	A.S	0009	A.TX	0001
ACIAT	0000	ADDBUF2	0426	ADDBUFNX	0416	ADDH	BFCE
ADDL	BFCC	ADDLEFTP	040B	ADR	BFCA	ASCIIL	E7EC
ASCIIR	E7F0	BUFLEFT	056A	CALLRDSO	012E	CANDATA	0126
DECBLEFT	0400	FWCOUNT	BEF7	INIT	ECFC	INTMASK	BEEE
INTPTMV	EBFB	IO	C000	LEFTPR	0566	LNAREADI	0545
LNATRANS	0544	LNBLW1	052F	LNBREADI	054C	LNBRW1	0539
LNBTRANS	054B	LNBUF	0526	LNCREADI	0553	LNCTRANS	0552
LNDREADI	055A	LNDTRANS	0559	LNEND	055F	LNFLW2	0534
LNFRW2	053E	LNFW	052C	LNSTART	0525	LNTIME	0527
LWCOUNT1	BEFD	LWCOUNT2	BEFF	MAININIT	0100	MEXOFF	0565
MORETIC	BEC7	MOVE1	E646	NXCHPR	0568	PTAB1	045D
PTAB1END	04B9	PTAB2	04BD	PTAB2END	04E9	PTAB3	04ED
PTAB3END	0509	PTAB4	050D	PTAB4END	0521	PTABNEXT	045B
PUT2HEX	0253	PUTA	02AA	PUTA2	02BA	PUTA3	02C4
PUTA4	02CE	PUTA5	02D8	PUTA6	02E2	PUTA7	02EC
PUTA8	02F0	PUTB	02FA	PUTB2	030A	PUTB3	0314
PUTB4	031E	PUTB5	0328	PUTB6	0332	PUTB7	033C
PUTB8	0340	PUTC	034A	PUTC2	035A	PUTC3	0364
PUTC4	036E	PUTC5	0378	PUTC6	0382	PUTC7	038C
PUTC8	0390	PUTD	039A	PUTD2	03AA	PUTD3	03B4
PUTD4	03BE	PUTD5	03C8	PUTD6	03D2	PUTD7	03DC
PUTD8	03E0	PUTFW	0262	PUTLINE	03EA	PUTLW	026C
PUTRW	028B	PUTTIME	0243	READSONA	EA54	RWCOUNT1	BEF9
RWCOUNT2	BEFB	SONCHANG	BEE1	SONDATRA	BEF0	SONDATRB	BEF2
SONDATRC	BEF4	SONDATRD	BEF6	SONDATSA	BEEF	SONDATSB	BEF1
SONDATSC	BEF3	SONDATSD	BEF5	SONEND	0447	SONEND2	044C
SONNEXA	BEE2	SONNEXB	BEE3	SONNEXC	BEE4	SONNEXD	BEE5
SONPRINT	ED5F	SONSW2	0453	SONSWITC	042D	SPBUFEND	9FFF
SPBUFNX	056C	SPBUFST	056E	SPBUFSTA	2000	TICTIME	BED9
TSTRX	019A	TSTRX2	01C5	TSTRX3	01AC	TSTRX4	01B7
TSTRX5	01C8	TSTRX6	01E5	TSTRX7	0202	TSTRX8	021F
TSTRX9	023C	TUSRAM	BC00	TWIDDLE	011E	UPNXCHPR	0189
XOFF	0013	XON	0011	XTEND	0121	XTEND2	0161
XTEND3	016C	XZERO	0000				

APPENDIX C

```
10 '
20 'DATE: 25 OCT 84
30 '
40 '############ M A P P E R #########
50 '
60 'THIS PROGRAM IS DESIGNED TO RUN ON A TRS-80 COLOR COMPUTER
70 'IT IS WRITTEN IN MICROSOFT DISK EXTENDED COLOR BASIC
80 'THE PURPOSE OF THIS PROGRAM IS TO TRANSFORM SONAR READINGS
85 'FROM THE AFIT ROBOT MARRS-1 INTO GRAPHICS FORMAT
90 'THE TEST ROOM IS 13' X 22' AND EACH PIXEL REPRESENTS 0.1 FT
100 'PROGRAM CONCEIVED AND WRITTEN BY CAPT HUBERT G. SCHNEIDER III
110 '
120 '
130 '
140 '
150 'MEMORY INITIALIZATION
160 CLEAR1000,&H6FFF:PCLEAR8:PMODE4,1:DEFUSR0 = &H7000
170 LOADM"ANDPAGE"
180 PMODE4,5:PCLS1:PMODE4,1:PCLS1
190 '
200 '
210 '
220 '
230 '
240 'GLOBAL DECLARATIONS
250 L1=0:L2=0:L3=0:L4=0:L5=0:R1=0:R2=0:R3=0:R4=0:R5=0:LW=0:RW=0
260 D=17.75:'DISTANCE BETWEEN REAR WHEELS IN INCHES
270 SN$="A0A4A1A5A2A6B0B4B1B5B2B6C0C4C1C5C2C6D0D4D1D5D2D6":' SONAR LOOKUP TABLE
280 C = 0:'COLOR
290 PI = 3.14159
300 '
310 'THE FOLLOWING FORMAT IS USED FOR THE DATA FILE:
320 '/time/fw/lw1/lw2/rw1/rw2/A#___/B#___/C#___/D#___/ <CR> <LF>
330 'WHERE time    = TENTH'S OF SECONDS COUNT - 2 BYTES
340 '      fw      = FRONT WHEEL DIRECTION    - 1 BYTE
350 '      lw#,rw# = REAR WHEEL COUNTS        - 2 BYTES EA.
360 '      A#___   = SONAR NUMBER AND READING - 4 BYTES EA.
370 '
380 'MAIN PROGRAM
390 '
400 '    GET INITIAL INFORMATION
410      GOSUB 1910
420 '
430 '    GET DRIVE NUMBERS
440      GOSUB1990
450 '
460 '    OPEN DATA FILE
470      GOSUB 2080
480 '
490 '    READ IN DATA
```

```
500 '    PCLS1
510      SCREEN1,1
520          FOR I=1TO6
530          SD$=""
540          LINEINPUT #1,SD$
550          IF EOF(1)=-1 THEN 720 :'CHECK FOR END OF FILE
560               FOR J=1TO11
570               P1=INSTR(1,SD$,"/")
580               A$=LEFT$(SD$,P1-1)
590               ON J GOSUB 840,870,930,960,1000,1040,1080,1500,1500,1500,1500
600               SD$=RIGHT$(SD$,LEN(SD$)-P1)
610               NEXT J
620          NEXT I
630      Q=USR0(0) :'CALL ML ROUTINE "ANDPAGE"
640 '
650 '    Z$=INKEY$:IFZ$=""THEN322     :' THESE  LINES ARE FOR
660 '    IF Z$="S" THEN 330 ELSE 360  :' THE OPTION OF SAVING
670 '    SAVE PICTURE                 :' EACH INCREMENTAL
680 '    F$=TIME$+".BIN:"+DN$         :' PICTURE OF WHAT THE
690 '    GOSUB1830                    :' SONARS "SEE"
700      GOTO 490 :'CONTINUE
710 '
720 '    CLOSE DATA FILE
730      GOSUB 2120
740 '
750 '    SAVE COMPOSITE PICTURE
760      FOR X= 5 TO 8:PCOPY X TO X-4:NEXT X:F$=T1$+"0COMP/BIN:"+DN$
770      GOSUB1830
780 '
790 END
800 '##########END OF PROGRAM##########
810 '
820 '############SUBROUTINES##########
830 '
840 'FIND FIRST DELIMITER IN DATA STRING ("/")
850 RETURN
860 '
870 'GET TIME
880 'TIC - COUNT FROM ROBOT
890 'TIME$ - DECIMAL COUNT
900 IF I=1 THEN TIME$=T1$+"0"+A$ ELSE RETURN
910 RETURN
920 '
930 'CONVERT FWCOUNT TO A VALUE
940 FW = VAL("&H"+A$) :RETURN
950 '
960 'CONVERT LWCOUNT1 TO A VALUE (LEFT WHEEL BACKWARDS)
970 LW=VAL("&H"+A$)
980 L1 = LW-L4:L4=LW:RETURN
990 '
1000 'CONVERT LWCOUNT2 TO A VALUE (LEFT WHEEL FORWARDS)
1010 LW=VAL("&H"+A$)
```

```
1020 L2 = LW-L5:L5=LW:RETURN
1030 '
1040 'CONVERT RWCOUNT1 TO A VALUE (RIGHT WHEEL BACKWARDS)
1050 RW=VAL("&H"+A$)
1060 R1= RW-R4:R4=RW:RETURN
1070 '
1080 'CONVERT RWCOUNT2 TO A VALUE (RIGHT WHEEL FORWARDS)
1090 RW=VAL("&H"+A$)
1100 R2 = RW-R5:R5=RW:GOSUB1110:PSET(X,Y,0):RETURN
1110 '
1120 'CALCULATE POISITION AND HEADING
1130 GOSUB1180 :'FIRST CALCULATE IN INCHES
1140 GOSUB1430 :'THEN CONERT TO GRAPHICS FORMAT
1150 XA=XB:YA=YB
1160 RETURN
1170 '
1180 '     DETERMINE HEADING & POSITION
1190 '     HEADING & POSITION ARE RELATIVE TO THE CENTER POINT BETWEEN THE WHEELS
1200 '     WHILE THE SONAR READINGS ARE RELATIVE TO THE CENTER OF THE ROBOT
1210 '     (XA,YA) - PREVIOUS POSITION
1220 '     (XB,YB) - CURRENT  POSITION
1230 '     HA & HB ARE IN RADIANS
1240 '     HA     - INITIAL  HEADING
1250 '     HB     - CURRENT  HEADING
1260 '     H      - GRAPHICS HEADING (0-1)
1270 '     D      - DISTANCE BETWEEN WHEELS ( 17.75" )
1280 '     *** NOTE ***
1290 '     FOR PURPOSES OF THIS TEST AND THESIS,
1300 '     THE UPPER LEVEL SONARS WILL REMAIN IN A FIXED POSITION
1310 '     DURING THE TEST (I.E. NO HEAD MOVEMENTS,
1320 '
1330 L3 = L2:R3 = R2
1340 DLR =L5-R5
1350 HB =(DLR/D) + HA
1360 HC=COS(HB):HS=SIN(HB)
1370 XB = XA +(((R3+L3)/2) * HC)
1380 YB = YA +(((R3+L3)/2) * HS)
1390 'NORMALIZE HEADING
1400 H = HB/(2*PI):IF H < 0 THEN H = 1 + H ELSE IF H=0 THEN H=1
1410 IF H>1 THEN H=H-1:GOTO 1410
1420 RETURN
1430 '     GET (X,Y)
1440 '     THIS ROUTINE CONVERTS THE (XB,YB) LOCATIONS INTO INTEGER NUMBERS
1445 '     SO THAT EACH PIXEL REPRESENTS 0.1 FT
1450 '     X OFFSET -> 17, Y OFFSET -> 30
1460 X=INT(((XB+(6*HC))*10)/12)+17
1470 Y=INT(((YB+(6*HS))*10)/12)+30
1480 RETURN
1490 '
1500 '     DISPLAY SONAR DATA
1510 A1$=LEFT$(A$,2):A2$=RIGHT$(A$,LEN(A$)-2)
1520 GOSUB1550:GOSUB1680:GOSUB1740
```

```
1530 RETURN
1540 '
1550 '     CALCULATE DIRECTION OF SONAR BEAM
1560 'A1$   - SONAR NUMBER
1570 'H - HEADING (0 - 1)
1580 'S1   - START ARC POSITION
1590 'S2   - STOP  ARC POSITION
1600 IF H=1 THEN H=0
1610 SN=INT((INSTR(1,SN$,A1$)/2))
1630 S1= -.025+(SN/24)+H
1640 S2=S1+ .05
1650 IF H=0 THEN H=1
1660 RETURN
1670 '
1680 '     CALCULATE RADIUS (RANGE OF TARGET)
1690 R= VAL(A2$)*10
1700 IF (R>75 OR R<18 OR R=255) THEN R = 0
1710 IF RIGHT$(A1$,1)="*" THEN R=0
1720 RETURN
1730 '
1740 '     DRAW SONAR ARC
1750 '     X  - X DIRECTION - GRAPHICS LOCATION
1760 '     Y  - Y DIRECTION - GRAPHICS LOCATION
1770 '     R  - RADIUS
1780 '     C  - COLOR
1790 IF S1<0 THEN CIRCLE(X,Y),R,C,1,S1+1,0:CIRCLE(X,Y),R,C,1,0,S2 ELSE CIRCLE(X,Y),R,C,1,S1,S2
1810 RETURN
1820 '
1830 '     SAVE SONAR PICTURE
1840 FR=FREE(DN):IF FR< 3 THEN 1850 ELSE 1870
1850 SOUND100,10:CLS:PRINT"     D I S K   F U L L  ! ! ! ! "
1855 PRINT"PLEASE CHANGE DISK  IN DRIVE    NUMBER "DN" AND PRESS ENTER TO  CONTINUE"
1860 Z$=INKEY$:IFZ$=""THEN1860 ELSE 1840
1870 SAVEM F$,&HE00,&H25FF,&H0000
1880 SCREEN1,1
1890 RETURN
1900 '
1910 '     GET INITIAL HEADING AND POSITION
1920 CLS:PRINT"PLEASE INPUT INITIAL HEADING ANDPOSITION OF ROBOT FOR THIS  TEST RUN."
1925 PRINT"POSITION IN TERMS OF (X,Y)"
1927 PRINT"WHERE X AND Y ARE TENTHS OF FEETAND HEADING IS BETWEEN  0 AND 1."
1930 INPUT"HEADING";H:HA=2*PI*H:IF HA=0 THEN HA=2*PI
1940 INPUT"POSITION X";X:INPUT"POSITION Y";Y:XA=1.2*X:YA=1.2*Y
1950 PR.IT"INPUT DATA FILENAME":PRINT:LINEINPUT T1$:T$=T1$+"/DAT"
1960 R$=MID$(T1$,3,1):' GET ROOM CONFIGURATION NUMBER
1970 RETURN
1980 '
1990 '     GET DRIVE NUMBERS
2000 CLS:PRINT"           WELCOME":PRINT"  TO THE LAB OF MAKE BELIEVE!"
2010 PRINT:PRINT:PRINT
2020 PRINT"WHICH DRIVE HAS THE SONAR/OSE   DATA IN IT?"
2030 A$=INKEY$:IF A$=""THEN 2030
```

```
2040 A=VAL(A$): IF A= 0 THEN DN = 1:DRIVE(0):DN$="1" ELSE IF A=1 THEN DN = 0:DRIVE(1):DN$="0" ELSE 2030
2050 PRINT:PRINT"          THANK YOU"
2060 RETURN
2070 '
2080 '     OPEN FILE
2090 OPEN"I",#1,T$
2100 RETURN
2110 '
2120 '     CLOSE FILE
2130 CLOSE #1
2140 RETURN
2150 '
2160 '
2170 '
2180 '
2190 '
2200 '****************************************************
2210 '*                                                  *
2220 '*         NAME: ANDPAGE                            *
2230 '*         WRITTEN IN 6809 MACHINE CODE             *
2240 '*         THIS ROUTINE MAKES A COMPOSITE OF M[illegible]RS-I  *
2250 '*         SONAR/OSE DATA BY ANDING ONE PA[illegible]APHICS  *
2260 '*         WITH ANOTHER.  THIS PROGRAM DES[illegible]O RUN   *
2270 '*         ON A TRS-80 COLOR COMPUTER               *
2280 '*                                                  *
2290 '****************************************************
2300 '          ORG     $7000
2310 '          PSHS    Y          *SAVE Y REGISTER
2320 '          LDX     #$E00      *START OF PAGE 1
2330 '          LDY     #$2600     *START OF PAGE 2
2340 ' LOOP     LDA     ,X+        *GET BYTE FROM PAGE 1
2350 '          ANDA    ,Y         *AND IT WITH BYTE FROM PAGE 2
2360 '          STA     ,Y+        *STORE BACK IN PAGE 2
2370 '          CMPX    #$2600     *DONE YET?
2380 '          BNE     LOOP       *IF NOT CONTINUE
2390 '          PULS    Y          *GET BACK Y
2400 '          RTS                *RETURN
2410 '          END
```


171

```
                        ;********************************************
                        ;*                                          *
                        ;*              APPENDIX  D                 *
                        ;*                                          *
                        ;********************************************
                        ;               FILE: NAVDEF.A
0000                            get navdef.a            ;NAV COMPUTER EQUATES AND SYS RAM USAGE
                        ;
                        ;       NAVDEF.A  DEFINES THE INPUT/OUTPUT AND SYSTEM RAM USAGE FOR
                        ;       THE MARRS-1 NAVIGATION COMPUTER
                        ;
BC00                    TUSRAM  EQU 0BC00H              ;TOP OF USER STACK RAM (STACKS GROW DOWNWARD)
BE00                    OSRAM   EQU TUSRAM+0200H        ;BASE ADDRESS FOR OPERATING SYSTEM RAM
BE00                    USRSTAK EQU OSRAM               ;USER STACK
BE7F                    BOS     EQU OSRAM+07FH          ;BOTTOM OF OPERATING SYSTEM STACK (BOS)
                        ;
                        ;
                        ;
                        ;       INPUT/OUTPUT EQUATE AREA
                        ;
                        ;
C000                    IO      EQU 0C000H              ;BASE ADDRESS FOR ALL INPUT/OUTPUT
                        ;
                        ;
0000                    ACIAT   EQU 0                   ;TERMINAL ACIA OFFSET FROM I/O BASE ADDRESS
0000                    A.T     EQU ACIAT
0008                    ACIAX   EQU 8                   ;XTRA ACIA OFFSET FROM I/O BASE ADDRESS
0008                    A.X     EQU ACIAX
0010                    ACIAL   EQU 16                  ;LASER ACIA OFFSET FROM I/O BASE ADDRESS
0010                    A.L     EQU ACIAL
                        ;
                        ;
0018                    PTMAB   EQU 24                  ;SONAR A&B TIMER OFFSET FROM I/O BASE ADDRESS
0018                    T.AB    EQU PTMAB
0020                    PTMCD   EQU 32                  ;SONAR C&D TIMER OFFSET FROM I/O BASE ADDRESS
0020                    T.CD    EQU PTMCD
0028                    PTMV    EQU 40                  ;VSW DRIVE/TIC TIMER OFFSET FROM I/O BASE ADDRESS
0028                    T.V     EQU PTMV
0030                    PTMF    EQU 48                  ;FRONT WHEEL TIMER OFFSET FROM I/O BASE ADDRESS
0030                    T.F     EQU PTMF
0038                    PTML    EQU 56                  ;LEFT WHEEL TIMER OFFSET FROM I/O BASE ADDRESS
0038                    T.L     EQU PTML
0040                    PTMR    EQU 64                  ;RIGHT WHEEL TIMER OFFSET FROM I/O BASE ADDRESS
0040                    T.R     EQU PTMR
                        ;
                        ;
0048                    PIAAB   EQU 72                  ;SONAR A&B TRANSDUCER SELECT PIA I/O OFFSET
0048                    P.AB    EQU PIAAB
0050                    PIACD   EQU 80                  ;SONAR C&D TRANSDUCER SELECT PIA I/O OFFSET
0050                    P.CD    EQU PIACD
                        ;
                        ;
```

```
0001    A0       EQU 1          ;MASK BIT TO SELECT SONAR TRANSDUCER A0
0002    A1       EQU 2          ;MASK BIT TO SELECT SONAR TRANSDUCER A1
0004    A2       EQU 4          ;MASK BIT TO SELECT SONAR TRANSDUCER A2
0008    A3       EQU 8          ;MASK BIT TO SELECT SONAR TRANSDUCER A3
0010    A4       EQU 16         ;MASK BIT TO SELECT SONAR TRANSDUCER A4
0020    A5       EQU 32         ;MASK BIT TO SELECT SONAR TRANSDUCER A5
0040    A6       EQU 64         ;MASK BIT TO SELECT SONAR TRANSDUCER A6
0080    A7       EQU 128        ;MASK BIT TO SELECT SONAR TRANSDUCER A7
0001    B0       EQU 1          ;MASK BIT TO SELECT SONAR TRANSDUCER B0
0002    B1       EQU 2          ;MASK BIT TO SELECT SONAR TRANSDUCER B1
0004    B2       EQU 4          ;MASK BIT TO SELECT SONAR TRANSDUCER B2
0008    B3       EQU 8          ;MASK BIT TO SELECT SONAR TRANSDUCER B3
0010    B4       EQU 16         ;MASK BIT TO SELECT SONAR TRANSDUCER B4
0020    B5       EQU 32         ;MASK BIT TO SELECT SONAR TRANSDUCER B5
0040    B6       EQU 64         ;MASK BIT TO SELECT SONAR TRANSDUCER B6
0080    B7       EQU 128        ;MASK BIT TO SELECT SONAR TRANSDUCER B7
0001    C0       EQU 1          ;MASK BIT TO SELECT SONAR TRANSDUCER C0
0002    C1       EQU 2          ;MASK BIT TO SELECT SONAR TRANSDUCER C1
0004    C2       EQU 4          ;MASK BIT TO SELECT SONAR TRANSDUCER C2
0008    C3       EQU 8          ;MASK BIT TO SELECT SONAR TRANSDUCER C3
0010    C4       EQU 16         ;MASK BIT TO SELECT SONAR TRANSDUCER C4
0020    C5       EQU 32         ;MASK BIT TO SELECT SONAR TRANSDUCER C5
0040    C6       EQU 64         ;MASK BIT TO SELECT SONAR TRANSDUCER C6
0080    C7       EQU 128        ;MASK BIT TO SELECT SONAR TRANSDUCER C7
0001    D0       EQU 1          ;MASK BIT TO SELECT SONAR TRANSDUCER D0
0002    D1       EQU 2          ;MASK BIT TO SELECT SONAR TRANSDUCER D1
0004    D2       EQU 4          ;MASK BIT TO SELECT SONAR TRANSDUCER D2
0008    D3       EQU 8          ;MASK BIT TO SELECT SONAR TRANSDUCER D3
0010    D4       EQU 16         ;MASK BIT TO SELECT SONAR TRANSDUCER D4
0020    D5       EQU 32         ;MASK BIT TO SELECT SONAR TRANSDUCER D5
0040    D6       EQU 64         ;MASK BIT TO SELECT SONAR TRANSDUCER D6
0080    D7       EQU 128        ;MASK BIT TO SELECT SONAR TRANSDUCER D7
        ;
        ;
0000    ACIASTAT EQU 0          ;OFFSET FROM ACIA BASE FOR STATUS REGISTER
0000    A.S      EQU ACIASTAT
0000    ACIACR   EQU 0          ;OFFSET FROM ACIA BASE FOR CONTROL REGISTER
0000    A.C      EQU ACIACR
0001    ACIATX   EQU 1          ;OFFSET FROM AICA BASE FOR TRANSMIT REGISTER
0001    A.TX     EQU ACIATX
0001    ACIARX   EQU 1          ;OFFSET FROM ACIA BASE FOR RECEIVE REGISTER
0001    A.RX     EQU ACIARX
        ;
        ;
0001    PTMSTAT  EQU 1          ;OFFSET FROM PTM BASE FOR STATUS REGISTER
0001    T.S      EQU PTMSTAT
        ;
        ;
0000    PTMCR1   EQU 0          ;OFFSET FROM PTM BASE FOR TIMER 1 CONTROL REGISTER
0000    T.C1     EQU PTMCR1
0001    PTMCR2   EQU 1          ;OFFSET FROM PTM BASE FOR TIMER 2 CONTROL REGISTER
0001    T.C2     EQU PTMCR2
```

```
0000                PTMCR3   EQU 0             ;OFFSET FROM PTM BASE FOR TIMER 3 CONTROL REGISTER
0000                T.C3     EQU PTMCR3
                    ;
                    ;
0002                T.LATCH1 EQU 2             ;PTM COUNTER LATCH FOR TIMER 1
0004                T.LATCH2 EQU 4             ;PTM COUNTER LATCH FOR TIMER 2
0006                T.LATCH3 EQU 6             ;PTM COUNTER LATCH FOR TIMER 3
                    ;
                    ;
0002                T.CNT1   EQU 2             ;PTM COUNTER COUNT FOR TIMER 1
0004                T.CNT2   EQU 4             ;PTM COUNTER COUNT FOR TIMER 2
0006                T.CNT3   EQU 6             ;PTM COUNTER COUNT FOR TIMER 3
                    ;
                    ;
0001                PIACRA   EQU 1             ;PIA A CONTROL REGISTER OFFSET FROM PIA BASE
0001                P.CRA    EQU PIACRA
0003                PIACRB   EQU 3             ;PIA B CONTROL REGISTER OFFSET FROM PIA BASE
0003                P.CRB    EQU PIACRB
0001                PIACRC   EQU 1             ;PIA C CONTROL REGISTER OFFSET FROM PIA BASE
0001                P.CRC    EQU PIACRC
0003                PIACRD   EQU 3             ;PIA D CONTROL REGISTER OFFSET FROM PIA BASE
0003                P.CRD    EQU PIACRD
                    ;
                    ;
0001                PIASTATA EQU 1             ;PIA A STATUS RECISTER OFFSET FROM PIA BASE
0001                P.SRA    EQU PIASTATA
0003                PIASTATB EQU 3             ;PIA B STATUS REGISTER OFFSET FROM PIA BASE
0003                P.SRB    EQU PIASTATB
0001                PIASTATC EQU 1             ;PIA C STATUS REGISTER OFFSET FROM PIA BASE
0001                P.SRC    EQU PIASTATC
0003                PIASTATD EQU 3             ;PIA D STATUS REGISTER OFFSET FROM PIA BASE
0003                P.SRD    EQU PIASTATD
                    ;
                    ;
0000                PIADDA   EQU 0             ;PIA A DATA DIRECTION REGISTER OFFSET FROM PIA BASE
0000                P.DDA    EQU PIADDA
0002                PIADDB   EQU 2             ;PIA B DATA DIRECTION REGISTER OFFSET FROM PIA BASE
0002                P.DDB    EQU PIADDB
0000                PIADDC   EQU 0             ;PIA C DATA DIRECTION REGISTER OFFSET FROM PIA BASE
0000                P.DDC    EQU PIADDC
0002                PIADDD   EQU 2             ;PIA D DATA DIRECTION REGISTER OFFSET FROM PIA BASE
0002                P.DDD    EQU PIADDD
                    ;
                    ;
0000                PIAPA    EQU 0             ;PIA A PERIPHERAL REGISTER OFFSET FROM PIA BASE
0000                P.PRA    EQU PIAPA
0002                PIAPB    EQU 2             ;PIA B PERIPHERAL REGISTER OFFSET FROM PIA BASE
0002                P.PRB    EQU PIAPB
0000                PIAPC    EQU 0             ;PIA C PERIPHERAL REGISTER OFFSET FROM PIA BASE
0000                P.PRC    EQU PIAPC
0002                PIAPD    EQU 2             ;PIA D PERIPHERAL REGISTER OFFSET FROM PIA BASE
0002                P.PRD    EQU PIAPD
```

```
                  ;
                  ;
                  ;
                  ;       SYSTEM RAM USAGE EQUATE AREA
                  ;
                  ;
BE80              CRACIAT  EQU OSRAM+080H       ;COPY OF CONTROL REGISTER BYTE FOR ACIAT
BE81              CRACIAX  EQU OSRAM+081H       ;COPY OF CONTROL REGISTER BYTE FOR ACIAX
BE82              CRACIAL  EQU OSRAM+082H       ;COPY OF CONTROL REGISTER BYTE FOR ACIAL
                  ;
                  ;
BE83              CRPTMAB1 EQU OSRAM+083H       ;COPY OF CONTROL REGISTER BYTE FOR PTMAB1
BE84              CRPTMAB2 EQU OSRAM+084H       ;COPY OF CONTROL REGISTER BYTE FOR PTMAB2
BE85              CRPTMAB3 EQU OSRAM+085H       ;COPY OF CONTROL REGISTER BYTE FOR PTMAB3
BE86              CRPTMCD1 EQU OSRAM+086H       ;COPY OF CONTROL REGISTER BYTE FOR PTMCD1
BE87              CRPTMCD2 EQU OSRAM+087H       ;COPY OF CONTROL REGISTER BYTE FOR PTMCD2
BE88              CRPTMCD3 EQU OSRAM+088H       ;COPY OF CONTROL REGISTER BYTE FOR PTMCD3
BE89              CRPTMV1  EQU OSRAM+089H       ;COPY OF CONTROL REGISTER BYTE FOR PTMV1
BE8A              CRPTMV2  EQU OSRAM+08AH       ;COPY OF CONTROL REGISTER BYTE FOR PTMV2
BE8B              CRPTMV3  EQU OSRAM+08BH       ;COPY OF CONTROL REGISTER BYTE FOR PTMV3
BE8C              CRPTMF1  EQU OSRAM+08CH       ;COPY OF CONTROL REGISTER BYTE FOR PTMF1
BE8D              CRPTMF2  EQU OSRAM+08DH       ;COPY OF CONTROL REGISTER BYTE FOR PTMF2
BE8E              CRPTMF3  EQU OSRAM+08EH       ;COPY OF CONTROL REGISTER BYTE FOR PTMF3
BE8F              CRPTML1  EQU OSRAM+08FH       ;COPY OF CONTROL REGISTER BYTE FOR PTML1
BE90              CRPTML2  EQU OSRAM+090H       ;COPY OF CONTROL REGISTER BYTE FOR PTML2
BE91              CRPTML3  EQU OSRAM+091H       ;COPY OF CONTROL REGISTER BYTE FOR PTML3
BE92              CRPTMR1  EQU OSRAM+092H       ;COPY OF CONTROL REGISTER BYTE FOR PTMR1
BE93              CRPTMR2  EQU OSRAM+093H       ;COPY OF CONTROL REGISTER BYTE FOR PTMR2
BE94              CRPTMR3  EQU OSRAM+094H       ;COPY OF CONTROL REGISTER BYTE FOR PTMR3
                  ;
                  ;
BE95              CRPIAA   EQU OSRAM+095H       ;COPY OF CONTROL REGISTER BYTE FOR PIAA
BE96              CRPIAB   EQU OSRAM+096H       ;COPY OF CONTROL REGISTER BYTE FOR PIAB
BE97              CRPIAC   EQU OSRAM+097H       ;COPY OF CONTROL REGISTER BYTE FOR PIAC
BE98              CRPIAD   EQU OSRAM+098H       ;COPY OF CONTROL REGISTER BYTE FOR PIAD
                  ;
                  ;
BE99              BCPTMAB1 EQU OSRAM+099H       ;BINARY COUNT PRELOAD FOR PTMAB1
BE9B              BCPTMAB2 EQU OSRAM+09BH       ;BINARY COUNT PRELOAD FOR PTMAB2
BE9D              BCPTMAB3 EQU OSRAM+09DH       ;BINARY COUNT PRELOAD FOR PTMAB3
BE9F              BCPTMCD1 EQU OSRAM+09FH       ;BINARY COUNT PRELOAD FOR PTMCD1
BEA1              BCPTMCD2 EQU OSRAM+0A1H       ;BINARY COUNT PRELOAD FOR PTMCD2
BEA3              BCPTMCD3 EQU OSRAM+0A3H       ;BINARY COUNT PRELOAD FOR PTMCD3
BEA5              BCPTMV1  EQU OSRAM+0A5H       ;BINARY COUNT PRELOAD FOR PTMV1
BEA7              BCPTMV2  EQU OSRAM+0A7H       ;BINARY COUNT PRELOAD FOR PTMV2
BEA9              BCPTMV3  EQU OSRAM+0A9H       ;BINARY COUNT PRELOAD FOR PTMV3
BEAB              BCPTMF1  EQU OSRAM+0ABH       ;BINARY COUNT PRELOAD FOR PTMF1
BEAD              BCPTMF2  EQU OSRAM+0ADH       ;BINARY COUNT PRELOAD FOR PTMF2
BEAF              BCPTMF3  EQU OSRAM+0AFH       ;BINARY COUNT PRELOAD FOR PTMF3
BEB1              BCPTML1  EQU OSRAM+0B1H       ;BINARY COUNT PRELOAD FOR PTML1
BEB3              BCPTML2  EQU OSRAM+0B3H       ;BINARY COUNT PRELOAD FOR PTML2
BEB5              BCPTML3  EQU OSRAM+0B5H       ;BINARY COUNT PRELOAD FOR PTML3
```

```
BEB7        BCPTMR1  EQU OSRAM+0B7H        ;BINARY COUNT PRELOAD FOR PTMR1
BEB9        BCPTMR2  EQU OSRAM+0B9H        ;BINARY COUNT PRELOAD FOR PTMR2
BEBB        BCPTMR3  EQU OSRAM+0BBH        ;BINARY COUNT PRELOAD FOR PTMR3
            ;
            ;
BEBD        SONARSLA EQU OSRAM+0BDH        ;SONAR SELECT BYTE WRITTEN TO PIA A
BEBD        SSLA     EQU SONARSLA
BEBE        SONARSLB EQU OSRAM+0BEH        ;SONAR SELECT BYTE WRITTEN TO PIA B
BEBE        SSLB     EQU SONARSLB
BEBF        SONARSLC EQU OSRAM+0BFH        ;SONAR SELECT BYTE WRITTEN TO PIA C
BEBF        SSLC     EQU SONARSLC
BEC0        SONARSLD EQU OSRAM+0C0H        ;SONAR SELECT BYTE WRITTEN TO PIA D
BEC0        SSLD     EQU SONARSLD
            ;
            ;
BEC1        SONARA   EQU OSRAM+0C1H        ;LAST SONAR A READING
BEC2        SONARB   EQU OSRAM+0C2H        ;LAST SONAR B READING
BEC3        SONARC   EQU OSRAM+0C3H        ;LAST SONAR C READING
BEC4        SONARD   EQU OSRAM+0C4H        ;LAST SONAR D READING
            ;
            ;
BEC5        SONTIME  EQU OSRAM+0C5H        ;TIC TIME OF LAST SONAR READING
            ;
BEC7        MORETIC  EQU OSRAM+0C7H        ;JUMP VECTOR FOR EXTENDED TIC INTERUPT ROUTINE
            ;
BECA        TIMEZERO EQU OSRAM+0CAH        ;TIME ZERO FROM DRIVE COMPUTER (15 BCD LS NIBBLES)
            ;
BED9        TICTIME  EQU OSRAM+0D9H        ;ONE TENTH SECOND TIC TIME COUNT SINCE TIME ZERO
            ;
BEDB        IOOFFSET EQU OSRAM+0DBH        ;OFFSET INTO I/O AREA FOR PORT INITIALIZATION
            ;
BEDC        POWERUP  EQU OSRAM+0DCH        ;SET TO $0 IF POWER UP NOT COMPLETE ELSE $55
            ;
BEDD        RAMSIZEL EQU OSRAM+0DDH        ;LOWER LIMIT OF RAM IN SYSTEM
            ;
BEDF        RAMSIZEH EQU OSRAM+0DFH        ;HIGH  LIMIT OF RAM IN SYSTEM
            ;
BEE1        SONCHANGE EQU OSRAM+0E1H       ;CHANGE SELECTED SONARS IF 00 ELSE INTERRUPT
                                           ;HANDLER SETS THIS BYTE TO 055H
BEE2        SONNEXA EQU OSRAM+0E2H         ;NEXT SONAR SELECT BYTE WRITTEN TO PIA A
BEE2        NSSLA    EQU SONNEXA
BEE3        SONNEXB EQU OSRAM+0E3H         ;NEXT SONAR SELECT BYTE WRITTEN TO PIA B
BEE3        NSSLB    EQU SONNEXB
BEE4        SONNEXC EQU OSRAM+0E4H         ;NEXT SONAR SELECT BYTE WRITTEN TO PIA C
BEE4        NSSLC    EQU SONNEXC
BEE5        SONNEXD EQU OSRAM+0E5H         ;NEXT SONAR SELECT BYTE WRITTEN TO PIA D
BEE5        NSSLD    EQU SONNEXD
BEE6        TICTEMP0 EQU OSRAM+0E6H        ;TEMPORARY REGISTER FOR TIC INTERRUPT
BEE8        TICTEMP1 EQU OSRAM+0E8H        ;TEMPORARY REGISTER FOR TIC INTERRUPT
BEEA        TICTEMP2 EQU OSRAM+0EAH        ;TEMPORARY REGISTER FOR TIC INTERRUPT
BEEC        TICTEMP3 EQU OSRAM+0ECH        ;TEMPORARY REGISTER FOR TIC INTERRUPT
            ;
```

```
BEEE            INTMASK  EQU OSRAM+0EEH          ;INTERRUPTS MASKED IF $FF ELSE NOT MASKED
                                                 ;IF $00
                ;
BEEF            SONDATSA EQU OSRAM+0EFH          ;SONAR SELECT BYTE FROM READSONAR ROUTINE
BEF0            SONDATRA EQU OSRAM+0F0H          ;SONAR READING FROM READSONAR ROUTINE
BEF1            SONDATSB EQU OSRAM+0F1H          ;SONAR SELECT BYTE FROM READSONAR ROUTINE
BEF2            SONDATRB EQU OSRAM+0F2H          ;SONAR READING FROM READSONAR ROUTINE
BEF3            SONDATSC EQU OSRAM+0F3H          ;SONAR SELECT BYTE FROM READSONAR ROUTINE
BEF4            SONDATRC EQU OSRAM+0F4H          ;SONAR READING FROM READSONAR ROUTINE
BEF5            SONDATSD EQU OSRAM+0F5H          ;SONAR SELECT BYTE FROM READSONAR ROUTINE
BEF6            SONDATRD EQU OSRAM+0F6H          ;SONAR READING FROM READSONAR ROUTINE
                ;
BEF7            OSEREG       EQU    OSRAM+0F7H       ;BEGIN ADDRESS OF WHEEL COUNTS
BEF7            FWCOUNT      EQU    OSEREG           ;FRONT WHEEL ABSOLUTE COUNT
BEF9            RWCOUNT1     EQU    OSEREG+2         ;RIGHT WHEEL PTM COUNTER 1 COUNTS
BEFB            RWCOUNT2     EQU    OSEREG+4         ;RIGHT WHEEL PTM COUNTER 2 COUNTS
BEFD            LWCOUNT1     EQU    OSEREG+6         ;LEFT  WHEEL PTM COUNTER 1 COUNTS
BEFF            LWCOUNT2     EQU    OSEREG+8         ;LEFT  WHEEL PTM COUNTER 2 COUNTS
                ;
                ;
                ;
BF01            OSEMOD       EQU    OSEREG+10        ;MODULO OF WHEEL COUNTS < 1°
                ;
                ;
                ;
BF01            RWMOD1       EQU    OSEMOD           ;RIGHT WHEEL PTM 1 MODULO COUNT
BF02            RWMOD2       EQU    OSEMOD+1         ;RIGHT WHEEL PTM 2 MODULO COUNT
BF03            LWMOD1       EQU    OSEMOD+2         ;LEFT  WHEEL PTM 1 MODULO COUNT
BF04            LWMOD2       EQU    OSEMOD+3         ;LEFT  WHEEL PTM 2 MODULO COUNT.
                ;
                ;
BF05            COLD     EQU    OSRAM+0105H      ;JUMP VECTOR FOR C (COLD START) COMMAND
BF08            WARM     EQU    OSRAM+0108H      ;JUMP VECTOR FOR W (WARM START) COMMAND
BF0B            ODESSEY  EQU    OSRAM+010BH      ;JUMP VECTOR FOR O (ODESSEY) COMMAND
                ;
                ;
                ;
                ;
                ;               EXPANSION AREA FOR EQUATES
                ;
                ;
                ;
BF7C            LKSONTMP EQU OSRAM+017CH         ;TEMPORARY REGISTER FOR SONAR PRINT ROUTINE
BF7E            LSXTEMP  EQU OSRAM+017EH         ;TEMPORARY REGISTER FOR SONAR PRINT ROUTINE
BF80            BUF      EQU OSRAM+0180H         ;LINE BUFFER START
BFC7            BUFEND   EQU OSRAM+01C7H         ;END LINE BUFFER
BFC8            OFFSET   EQU OSRAM+01C8H
BFCA            ADR      EQU OSRAM+01CAH
BFCC            ADDL     EQU OSRAM+01CCH
BFCE            ADDH     EQU OSRAM+01CEH
BFD0            BUFPTR   EQU OSRAM+01D0H
BFD2            RECTYP   EQU OSRAM+01D2H
```

```
BFD3        COUNT    EQU OSRAM+01D3H
BFD4        CKSUM    EQU OSRAM+01D4H
BFD5        TEMP0    EQU OSRAM+01D5H     ;TEMPORARY REGISTERS MAYNOT BE USED
BFD7        TEMP1    EQU OSRAM+01D7H     ;BY ROUTINES THAT ARE ASSOCIATED WITH
BFD9        TEMP2    EQU OSRAM+01D9H     ;INTERRUPTS
BFDB        TEMP3    EQU OSRAM+01DBH
BFDD        TEMP4    EQU OSRAM+01DDH
BFDF        TEMP5    EQU OSRAM+01DFH
BFE1        TEMP6    EQU OSRAM+01E1H
BFE3        TEMP7    EQU OSRAM+01E3H
BFE5        MYFLAG0  EQU OSRAM+01E5H     ;TBD
BFE6        MYFLAG1  EQU OSRAM+01E6H     ;TBD
BFE7        SYSFLAG  EQU OSRAM+01E7H     ;TBD
BFE8        ECHO     EQU OSRAM+01E8H
BFE9        TCOUNT   EQU OSRAM+01E9H
BFEA        CREG     EQU OSRAM+01EAH
BFEB        BREG     EQU OSRAM+01EBH
BFEC        AREG     EQU OSRAM+01ECH
BFED        XREG     EQU OSRAM+01EDH
BFEF        PREG     EQU OSRAM+01EFH
BFF1        SREG     EQU OSRAM+01F1H
BFF3        USWI     EQU OSRAM+01F3H
BFF5        ACIAI    EQU OSRAM+01F5H
BFF7        IRQVEC   EQU OSRAM+01F7H
BFFA        SWIVEC   EQU OSRAM+01FAH
BFFD        NMIVEC   EQU OSRAM+01FDH
A000        MIOV     EQU 0A000H
A002        MBEGA    EQU 0A002H
A004        MENDA    EQU 0A004H
A005        MENDA1   EQU 0A005H
A006        MNIO     EQU 0A006H
A008        MSP      EQU 0A008H
A00A        MCKSM    EQU 0A00AH
A00B        MBYTECT  EQU 0A00BH
A00C        MXHI     EQU 0A00CH
A00D        MXLOW    EQU 0A00DH
A00E        MTEMP    EQU 0A00EH
A00F        MTW      EQU 0A00FH
A010        MTW1     EQU 0A010H
A011        MKCUNT   EQU 0A011H
A012        MXTEMP   EQU 0A012H
A042        MSTACK   EQU 0A042H
0000        MACIA    EQU ACIAT
BFF5        MACIAI   EQU ACIAI
            ;
            ;        THIS AREA FOR SPECIAL SYMBOLIC EQUATES
            ;
0000        XZERO    EQU 0000H           ;LOADS ZERO INTO INDEX REGISTER
            ;
0000        END

No error(s).
```

SYMBOL TABLE FOR FILE NAVDEF.A

A.C	0000	A.L	0010	A.RX	0001	A.S	0000
A.T	0000	A.TX	0001	A.X	0008	A0	0001
A1	0002	A2	0004	A3	0008	A4	0010
A5	0020	A6	0040	A7	0080	ACIACR	0000
ACIAI	BFF5	ACIAL	0010	ACIARX	0001	ACIASTAT	0000
ACIAT	0000	ACIATX	0001	ACIAX	0008	ADDH	BFCE
ADDL	BFCC	ADR	BFCA	AREG	BFEC	B0	0001
B1	0002	B2	0004	B3	0008	B4	0010
B5	0020	B6	0040	B7	0080	BCPTMAB1	BE99
BCPTMAB2	BE9B	BCPTMAB3	BE9D	BCPTMCD1	BE9F	BCPTMCD2	BEA1
BCPTMCD3	BEA3	BCPTMF1	BEAB	BCPTMF2	BEAD	BCPTMF3	BEAF
BCPTML1	BEB1	BCPTML2	BEB3	BCPTML3	BEB5	BCPTMR1	BEB7
BCPTMR2	BEB9	BCPTMR3	BEBB	BCPTMV1	BEA5	BCPTMV2	BEA7
BCPTMV3	BEA9	BOS	BE7F	BREG	BFEB	BUF	BF80
BUFEND	BFC7	BUFPTR	BFD0	C0	0001	C1	0002
C2	0004	C3	0008	C4	0010	C5	0020
C6	0040	C7	0080	CKSUM	BFD4	COLD	BF05
COUNT	BFD3	CRACIAL	BE82	CRACIAT	BE80	CRACIAX	BE81
CREG	BFEA	CRPIAA	BE95	CRPIAB	BE96	CRPIAC	BE97
CRPIAD	BE98	CRPTMAB1	BE83	CRPTMAB2	BE84	CRPTMAB3	BE85
CRPTMCD1	BE86	CRPTMCD2	BE87	CRPTMCD3	BE88	CRPTMF1	BE8C
CRPTMF2	BE8D	CRPTMF3	BE8E	CRPTML1	BE8F	CRPTML2	BE90
CRPTML3	BE91	CRPTMR1	BE92	CRPTMR2	BE93	CRPTMR3	BE94
CRPTMV1	BE89	CRPTMV2	BE8A	CRPTMV3	BE8B	D0	0001
D1	0002	D2	0004	D3	0008	D4	0010
D5	0020	D6	0040	D7	0080	ECHO	BFE8
FWCOUNT	BEF7	INTMASK	BEEE	IO	C000	IOOFFSET	BEDB
IRQVEC	BFF7	LKSONTMP	BF7C	LSXTEMP	BF7E	LWCOUNT1	BEFD
LWCOUNT2	BEFF	LWMOD1	BF03	LWMOD2	BF04	MACIA	0000
MACIAI	BFF5	MBEGA	A002	MBYTECT	A00B	MCKSM	A00A
MENDA	A004	MENDA1	A005	MIOV	A000	MMCONT	A011
MNIO	A006	MORETIC	BEC7	MSP	A008	MSTACK	A042
MTEMP	A00E	MTW	A00F	MTW1	A010	MXHI	A00C
MXLOW	A00D	MXTEMP	A012	MYFLAG0	BFE5	MYFLAG1	BFE6
NMIVEC	BFFD	NSSLA	BEE2	NSSLB	BEE3	NSSLC	BEE4
NSSLD	BEE5	ODESSEY	BF0B	OFFSET	BFC8	OSEMOD	BF01
OSEREG	BEF7	OSRAM	BE00	P.AB	0048	P.CD	0050
P.CRA	0001	P.CRB	0003	P.CRC	0001	P.CRD	0003
P.DDA	0000	P.DDB	0002	P.DDC	0000	P.DDD	0002
P.PRA	0000	P.PRB	0002	P.PRC	0000	P.PRD	0002
P.SRA	0001	P.SRB	0003	P.SRC	0001	P.SRD	0003
PIAAB	0048	PIACD	0050	PIACRA	0001	PIACRB	0003
PIACRC	0001	PIACRD	0003	PIADDA	0000	PIADDB	0002
PIADDC	0000	PIADDD	0002	PIAPA	0000	PIAPB	0002
PIAPC	0000	PIAPD	0002	PIASTATA	0001	PIASTATB	0003
PIASTATC	0001	PIASTATD	0003	POWERUP	BEDC	PREG	BFEF
PTMAB	0018	PTMCD	0020	PTMCR1	0000	PTMCR2	0001
PTMCR3	0000	PTMF	0030	PTML	0038	PTMR	0040
PTMSTAT	0001	PTMV	0028	RAMSIZEH	BEDF	RAMSIZEL	BEDD

RECTYP	BFD2	RWCOUNT1	BEF9	RWCOUNT2	BEFB	RWMOD1	BF01
RWMOD2	BF02	SONARA	BEC1	SONARB	BEC2	SONARC	BEC3
SONARD	BEC4	SONARSLA	BEBD	SONARSLB	BEBE	SONARSLC	BEBF
SONARSLD	BEC0	SONCHANG	BEE1	SONDATRA	BEF0	SONDATRB	BEF2
SONDATRC	BEF4	SONDATRD	BEF6	SONDATSA	BEEF	SONDATSB	BEF1
SONDATSC	BEF3	SONDATSD	BEF5	SONNEXA	BEE2	SONNEXB	BEE3
SONNEXC	BEE4	SONNEXD	BEE5	SONTIME	BEC5	SREG	BFF1
SSLA	BEBD	SSLB	BEBE	SSLC	BEBF	SSLD	BEC0
SWIVEC	BFFA	SYSFLAG	BFE7	T.AB	0018	T.C1	0000
T.C2	0001	T.C3	0000	T.CD	0020	T.CNT1	0002
T.CNT2	0004	T.CNT3	0006	T.F	0030	T.L	0038
T.LATCH1	0002	T.LATCH2	0004	T.LATCH3	0006	T.R	0040
T.S	0001	T.V	0028	TCOUNT	BFE9	TEMP0	BFD5
TEMP1	BFD7	TEMP2	BFD9	TEMP3	BFDB	TEMP4	BFDD
TEMP5	BFDF	TEMP6	BFE1	TEMP7	BFE3	TICTEMP0	BEE6
TICTEMP1	BEE8	TICTEMP2	BEEA	TICTEMP3	BEEC	TICTIME	BED9
TIMEZERO	BECA	TUSRAM	BC00	USRSTAK	BE00	USWI	BFF3
WARM	BF08	XREG	BFED	XZERO	0000		

```
;*******************************************************
;*                                                     *
;*          APPENDIX  E                                *
;*                                                     *
;*******************************************************
;         FILE: DRIVEDEF.A
;
;         MARRS-1 DRIVE COMPUTER DEFINITIONS
;
;         THE DRIVE COMPUTER USES THE VIRTUAL DEVICES
;
;     MENOS UPGRADE FOR THE HEATH HERO COMPUTER.  AS SUCH,
;
;     THE CPU IS A MOTOROLA 6801 AND SYSTEM RAM USAGE IS
;
;     SLIGHTLY DIFFERENT FROM THE STANDARD HERO COMPUTER.
;
;     THE BAUD RATE FOR THE 6801 SERIAL COMMUNICATION INTERFACE
;
;     IS 300 BAUD WHEN POWERING UP IN LILBUG MODE AND 9600
;
;     BAUD WHEN IN MENOS MODE.  THE ORIGINAL BAUD RATE
;
;     FOR MENOS WAS 300.  THE 9600 BAUD RATE WAS OBTAINED BY
;
;     BURNING A NEW ROM (U301) FOR THE MENOS UPGRADE BOARD
;
;     AND CHANGING THE FOLLOWING ROM LOCATIONS
;
;         0FEE9H      0FF06H      0FF91H
;
;     WITH A NEW BAUD RATE DIVISOR BYTE:
;
;         05      FOR 9600 BAUD
;
;         06      FOR 1200 BAUD
;
;         07      FOR  300 BAUD
;
;
0000  ;     ORG     0000H

;RAM ALLOCATIONS FOR MODIFIED HERO (0000-03FF)
;
;
;Version 1.1C.
;Written by A. H. Ballard 10/22/83
;Version 2.
;Written by A. H. Ballard 1/14/84
;
;
```

```
                ;   The original HERO ROM monitor (version 1.0) made use
                ;of the first 63 bytes of the 4K RAM for status data and
                ;jump vectors, and the top 287 bytes of RAM for stacks
                ;and scratchpad areas.  The central 3746 bytes (003F to
                ;0EE0) were left available for user programs.  Version
                ;1.1 of the HERO ROM used another 2 bytes in high RAM.
                ;The modified ROM produced by Virtual Devices reserves
                ;the first 32 bytes in ROM for use as control and status
                ;registers for a new 6801 microprocessor. The original
                ;63 bytes in RAM were moved up by 32 bytes (20 Hex added
                ;to all addresses).  In addition, the Virtual Devices
                ;modification (Version 10) uses 2 bytes in high RAM as a
                ;buffer for serial communications.
                ;
                ;   Immediately after a cold start or a warm restart,
                ;certain of these RAM locations are initialized to
                ;default values.  See the file MHINI1.A for details
                ;of RAM initialization in the HERO robot, as modified by
                ;Virtual Devices.
                ;
                ;
                ;
                ;Current overall allocation of RAM:
                ;
                ;  Low RAM used by Mod HERO      95 bytes (0000-005E)
                ;  User memory available       3710 bytes (005F-0EDC)
                ;  High RAM used by Mod HERO    291 bytes (0EDD-03FF)
                ;
                ;Reserved bytes in low RAM:
                ;
                ;
                ;6801 control and status registers:
                ;
0000            DDR1    RMB 1   ;Port 1 data direction register
                                ;
                                ;  0 = input
                                ;  1 = output
                ;
0001            DDR2    RMB 1   ;Port 2 data direction register
                                ;
                                ;  0 = input
                                ;  1 = output
                ;
                ;
0002            DR1     RMB 1   ;Port 1 data register
                                ;
                                ;  Bits 7,6 = select memory map
                                ;
                                ;    00 = (not used)
                                ;    01 = SYSROM $6000, SYSRAM $E000
                                ;    10 = SYSRAM $6000, HROM $E000
                                ;    11 = SYSRAM $6000, SYSROM $E000
```

```
                              ;
                              ;  Bit 5 = green LED (0/1 = OFF/ON)
                              ;  Bit 4 = select memory bank 0/1
                              ;  Bit 3 = radio (0/1 = RCV/XMT)
                              ;  Bit 2 = timer 3 output
                              ;  Bit 1 = timer 2 output
                              ;  Bit 0 = timer 2 input
                  ;
0003              DR2     RMB 1   ;Port 2 data register
                              ;
                              ;  Bits 7,6,5 = 6801 mode (R/O)
                              ;  Bit 4 = serial comm output
                              ;  Bit 3 = serial comm input
                              ;  Bit 2 = clock for serial comm
                              ;  Bit 1 = timer 1 output
                              ;  Bit 0 = timer 1 input
                  ;
                  ;
                  ;
                  ;
0004                      ORG $0004
                  ;
0004              DDR3    RMB 1   ;Port 3 data direction register (not used)

0004              DR2CPY  EQU $04         ; COPY OF Port 2 data register
                  ;           ^^^^^
                  ;           !!!!!    CRITICAL ADDRESS!!!!!!!!!!!!
                  ;
                  ;
                  ;
0005                      ORG $0005
                  ;
                  ;
0005              DDR4    RMB 1   ;Port 4 data direction register
                              ;
                              ;  Set 1's for address output
                  ;
                  ;
                  ;
0006                      ORG $0006
                  ;
0006              DR3     RMB 1   ;Port 3 data register (not used)
                  ;
0006              DDR1CPY EQU $06         ; COPY OF Port 1 data direction register
                  ;
                  ;
                  ;
                  ;
0007                      ORG $0007
                  ;
0007              DR4     RMB 1   ;Port 4 data register (not used)
                  ;
```

```
                    ;
                    ;
0008                       ORG $0008
                    ;
0008                TCSR   RMB 1   ;Timer 1 control and status register
                    ;
                    ;  Bit 7 = ICF, input capture flag (R/O)
                    ;  Bit 6 = OCF, output compare flag (R/O)
                    ;  Bit 5 = TOF, timer overflow flag (R/O)
                    ;  Bit 4 = EICI, enable input capture interrupts
                    ;  Bit 3 = EOCI, enable output compare interrupts
                    ;  Bit 2 = ETOI, enable timer overflow interrupts
                    ;  Bit 1 = IEDG, input edge polarity (0=HL, 1=LH)
                    ;  Bit 0 = OLVL, output level select (0=L, 1=H)
                    ;
0009                CNTR   RMB 2   ;16-bit CPU counter register (R/O)
                    ;
                    ;
000B                OCR1   RMB 2   ;16-bit output compare register 1
                    ;
                    ;
000D                ICR1   RMB 2   ;16-bit input capture register 1 (R/O)
                    ;
                    ;
                    ;
000F                       ORG $000F
                    ;
000F                CR3    RMB 1   ;Port 3 control register (not used)
000F                DR1CPY EQU $0F         ; COPY OF Port 1 data register
                    ;
                    ;
0010                       ORG $0010
                    ;
0010                RMCR   RMB 1   ;SCI rate and mode control register
                    ;
                    ;  Bits 7-4 = (not used)
                    ;
                    ;  Bits 3,2 = format and clock control
                    ;
                    ;    00 = Bi-phase, internal clock, P22 not used
                    ;    01 = NRZ, internal clock, P22 not used
                    ;    10 = NRZ, internal clock, P22 is clock output
                    ;    11 = NRZ, external clock, P22 is clock input
                    ;
                    ;  Bits 1,0 = baud rate select
                    ;
                    ;    00 = 76800 bps
                    ;    01 =  9600 bps
                    ;    10 =  1200 bps
                    ;    11 =   300 bps
                    ;
```

```
0011            TRCS    RMB 1   ;SCI transmit/receive control/status
                ;
                ;  Bit 7 = RDRF, receive data register full flag (R/O)
                ;  Bit 6 = ORFE, overrun/framing error flag (R/O)
                ;  Bit 5 = TDRE, transmit data register empty flag (R/O)
                ;  Bit 4 = RIE, receiver interrupt enalbe
                ;  Bit 3 = RE, receiver enable
                ;  Bit 2 = TIE, transmitter interrupt enable
                ;  Bit 1 = TE, transmitter enable
                ;  Bit 0 = WU, enable wake-up on next message
                ;
0012            RDR     RMB 1   ;SCI receive data register
                ;
                ;
0013            TDR     RMB 1   ;SCI transmit data register
                ;
                ;
0014            RAMC    RMB 1   ;RAM control register
                ;
                ;  Bit 7 = STBY, set to monitor voltage (0/1=LO/HI)
                ;  Bit 6 = RAME, 0/1=enable/disable on-chip RAM ($80FF)
                ;  Bits 5-0 = (not used)
                ;
                ;
                ;
                ;
0015                    ORG $0015
                ;
                ;Additional registers for 6801U4 version:
                ;
                ; THE FOLLOWING ARE NOT IMPLEMENTED IN THE BASIC 6801 OR 6301
                ; THESE ARE BEING RESERVED FOR THE POTENTIAL USE OF THE 6801U4
                ;
0015            ACNT    RMB 2   ;Alternate counter address
0017            TCR1    RMB 1   ;Timer control register 1
0018            TCR2    RMB 1   ;Timer control register 2
0019            TSR     RMB 1   ;Timer status register
001A            OCR2    RMB 2   ;Output compare register 2
001C            OCR3    RMB 2   ;Output compare register 3
001E            ICR2    RMB 2   ;Input capture register 2
                ;
                ;
                ;
                ;
0020                    ORG $0020
                ;
```

```
                ;HERO status data:
                ;
0020            mpos    EQU *
0020            EXTPOS  RMB 1   ;Current position of arm extend motor
0021            SHLPOS  RMB 1   ;Current position of shoulder motor
0022            ROTPOS  RMB 1   ;Current position of wrist rotate motor
0023            PVTPOS  RMB 1   ;Current position of wrist pivot motor
0024            GRPPOS  RMB 1   ;Current position of gripper motor
0025            HEDPOS  RMB 1   ;Current position of head motor
0026            STRPOS  RMB 1   ;Current position of steering motor
0027            DRVPOS  RMB 2   ;Spoke count for base motor command
0029            DRVODM  RMB 2   ;Odometer spoke count for base motor
                ;
                ;
002B            ACTSTAT RMB 1   ;Activity status byte
                                ;
                                ;  Bits 7-5 = (not used)
                                ;  Bit 4 = base speed ramping
                                ;  Bit 3 = speech active
                                ;  Bit 2 = arm/head motors active
                                ;  Bit 1 = steering motor active
                                ;  Bit 0 = base motor active

                ;
002C            ABTSTAT RMB 1   ;Abort request status byte
                                ;
                                ;  Bits 7-4 = (not used)
                                ;  Bit 3 = abort speech
                                ;  Bit 2 = abort arm/head motors
                                ;  Bit 1 = abort steering motor
                                ;  Bit 0 = abort base motor
                ;
002D            SLEEPER RMB 2   ;Sleep count, 10 sec steps
002F            HUNGRY  RMB 1   ;Low (logic) battery indicator byte
0030            SNRHIT  RMB 1   ;Cum count for detected sonar echoes
0031            SNRRNG  RMB 1   ;Most recently measured sonar range
                ;
                ;
                ;
                ;
0032                    ORG $0032
                ;
                ;User-supplied jump vectors:
                ;
                ;  Default (R) = RTS;  (SR) = SEC,RTS;  (-) = none
                ;
0032            USRCYCLE RMB 3  ;Customize start of interp cycle    (R)
0035            USRINTRP RMB 3  ;Extend interp command list        (SR)
0038            USRIRQ   RMB 3  ;Customize IRQ handler              (R)
003B            USRCLOCK RMB 3  ;Customize clock IRQ routine        (R)
003E            USRSPD   RMB 3  ;Customize drive speed routine      (R)
0041            USRLLB   RMB 3  ;New IRQ, low logic battery        (SR)
```

```
0044            USRLDB   RMB 3  ;New IRQ, low drive battery       (SR)
0047            USRMDT   RMB 3  ;User program for motion detector   (R)
004A            USRTRG   RMB 3  ;User program for pendant trigger   (R)
004D            USREXP   RMB 3  ;User IRQ program for exper board   (R)
0050            USER1    RMB 3  ;Jump vector for Key 9 in exec mode (-)
0053            USER2    RMB 3  ;Jump vector for Key C in exec mode (-)
0056            USER3    RMB 3  ;Jump vector for Key F in exec mode (-)
                ;
                ;
                ;
                ;
                ;       ORG $0059
                ;
                ;Scratch pad registers:
                ;
0059            T0      RMB 2   ;Used in utility, manual, learn modes
005B            T1      RMB 2   ;Used in utility, manual, learn modes
005D            T2      RMB 2   ;Used in utility, manual, learn modes
                ;
                ;
                ;
                ;
                ;
005F                    ORG $005F
                ;
                ;
                ;Ram area available for user programs:
                ;
005F            USRRAM  RMB 3710
                ;
                ;
                ;
                ;
0EDD                    ORG     $0EDD       ;HIGH HERO RAM
                ;
                ;Reserved bytes in high RAM:
                ;
                ;Used only by Virtual Devices modifications:
                ;
                ;RXSTAT EQU $0EDD        ;FF = good byte, 00 = none/bad
0EDD            RXSTAT  RMB 1            ;00=no/bad byte, FF=good byte
                ;RXBUFF EQU $0EDE        ;Holds good byte received
0EDE            RXBUFF  RMB 1            ;Holds good byte received
                ;
                ;Used only in HERO 1.1 and later versions:
                ;
0EDF            PWRTEMP RMB 2   ;AAAA is put here when re-initializing
                                ;to home CPOS registers but not motors
                ;
                ;
                ;Used in executive/interpreter modes:
                ;
```

```
0EE1            MODE     RMB 1  ;00=native mode, FF=robot mode
                ;
0EE2            PCSAVE   RMB 2  ;User program counter stored here
                ;
                ;
0EE4            STRSTP   RMB 12 ;Buffer area for steer stepper motor
0EF0            ARMSTP   RMB 12 ;Buffer area for arm/head steppers
                ;
                ;  Format for STRSTP and ARMSTP buffers:
                ;
                ;    Byte 0 = STPMASK, mask for output port
                ;
                ;      $0F = low nibble
                ;      $F0 = high nibble
                ;
                ;    Byte 1 = STPSCL, scale factor
                ;
                ;      $03 =  4 physical steps per logical step
                ;      $07 =  8 physical steps per logical step
                ;      $0F = 16 physical steps per logical step
                ;      $1F = 32 physical steps per logical step
                ;
                ;    Bytes 2,3 = STPPORT, stepper output port
                ;
                ;    Bytes 4,5 = STPPOS, address of current position
                ;
                ;    Byte 6 = STPLIM, limit switch mask for port $C260
                ;
                ;      $24 for shoulder motor
                ;      $12 for head motor
                ;      $48 for steering motor
                ;      $00 for other motors
                ;
                ;    Byte 7 = STPTIME, speed divisor (decremented to 0)
                ;
                ;    Byte 8 = STPTIME0, speed divisor (fixed)
                ;
                ;    Byte 9 = STPSEQ, step pattern sequence select
                ;
                ;      $00-03 for forward (start at $00)
                ;      $04-07 for reverse (start at $04)
                ;
                ;    Bytes 10,11 = STPNO, number of physical steps to go
                ;
                ;
                ;
                ;
0EFC            CLCKTIME RMB 1  ;8-bit count for 1024 Hz clock
                ;
0EFD            DRVNO    RMB 2  ;Steps to go (DPOS) for base motor
                ;
0EFF            STRPCPY  RMB 1  ;Last byte sent to STRPORT=C260
```

```
0F00            DRVPCPY  RMB 1  ;Last byte sent to DRVPORT=C2A0
0F01            DRVPNEW  RMB 1  ;New byte to go to DRVPORT=C2A0
0F02            ADRPCPY  RMB 1  ;Last byte sent to ADRPORT=C2C0
0F03            PWRPCPY  RMB 1  ;Last byte sent to PWRPORT=C2E0
0F04            IRQPCPY  RMB 1  ;Last byte sent to IRQPORT=C220
                ;
0F05            SPSKAREA RMB 16 ;Speech stack, max of 8 addresses
0F15            SPSKFWA  RMB 2  ;Location, 1st speech stack entry
0F17            SPSKLWA  RMB 2  ;Location, 8th speech stack entry
0F19            SPSKPTR  RMB 2  ;Speech stack pointer saved here
                ;
0F1B            MTRSPD   RMB 1  ;Base motor speed byte stored here
0F1C            STRCTADR RMB 2  ;STRSTP/ARMSTP selection saved here
0F1E            MCB      RMB 2  ;Buffer for motor control bytes
0F20            ACTMASK  RMB 1  ;Mask bits for ACTSTAT byte
0F21            ACTIVATE RMB 1  ;Buffer for new ACTSTAT bits
0F22            DEST     RMB 2  ;Destination address for block move
0F24            SRC      RMB 2  ;Source address for block move
0F26            MVALLOP  RMB 2  ;Buffer for address of 1st MVALL operand
                ;
                ;
                ;
                ;
                ;Used in learn/manual mode only:
                ;
0F28            REVPOS   RMB 2  ;
0F2A            REVADDR  RMB 2  ;
0F2C            ADDRPTR  RMB 2  ;
0F2E            ADDR     RMB 2  ;
0F30            OLDADDR  RMB 2  ;
0F32            FRSTADDR RMB 2  ;
0F34            LASTADDR RMB 2  ;
0F36            LRNMODE  RMB 1  ;
0F37            OLDSLCT  RMB 1  ;
0F38            OLDDIR   RMB 1  ;
0F39            OLDPOS   RMB 1  ;
0F3A            ISOLD    RMB 1  ;
0F3B            SLCT     RMB 1  ;
0F3C            DIR      RMB 1  ;
0F3D            ISARM    RMB 1  ;
0F3E            DRVON    RMB 1  ;
0F3F            STRGON   RMB 1  ;
0F40            ARMON    RMB 1  ;
0F41            ARMBACK  RMB 1  ;
0F42            ARMSTOP  RMB 1  ;
0F43            DELAYTM  RMB 2  ;
0F45            TRIGIN   RMB 1  ;
0F46            BASEOP   RMB 1  ;
0F47            OPCODE   RMB 1  ;
0F48            POS      RMB 2  ;
0F4A            MTR      RMB 1  ;
0F4B            SPD      RMB 1  ;
```

```
0F4C            OLDDIS   RMB 2  ;
0F4E            DISLMDE  RMB 1  ;
0F4F            OLDLMDE  RMB 1  ;
0F50            XTEMP    RMB 2  ;
                ;
                ;Used by IRQ handler only:
                ;
0F52            STPPTR   RMB 2  ;
0F54            TEMP1    RMB 2  ;
0F56            SPSAVE   RMB 2  ;
                ;
                ;Used in executive and interpreter modes
                ;
0F58            USRSTK   RMB 64 ;User stack area
0F98            MONSTK   RMB 64 ;Monitor stack area
0FD8            PWRSTK   RMB 8  ;Power-on stack area
0FE0            BKTBL    RMB 12 ;Table of 4 breakpoint vectors
0FEC            BKPNTSTK RMB 2  ;Pointer for breakpoint stack
0FEE            TE0      RMB 2  ;Temporary register, exec mode
0FF0            TE1      RMB 2  ;Temporary register, exec mode
0FF2            DIGADD   RMB 2  ;Address of next LED display digit
0FF4            USERS    RMB 2  ;User's stack pointer
0FF6            SYSSWI   RMB 3  ;Software interrupt vector
0FF9            DSHPTR   RMB 2  ;Digit address for moving dash display
0FFB            DSHDIR   RMB 1  ;Direction of moving dash display
0FFC            TIME1    RMB 2  ;Display time for "HERO 1.x"
0FFE            TIME2    RMB 2  ;Display time for moving dash
                ;
                ;Used in executive and interpreter modes:
                ;
                ;
                ;End of 4K RAM
                ;

                ;
                ; CONTROL AND STATUS PORTS - - MEMORY MAPPED PHYSICAL ADDRESSES
                ;
C220            SNRTIMR    EQU $C220   ;SONAR TIMER
C240            SENSE      EQU $C240   ;SENSE INPUT
C260            LIMIT      EQU $C260   ;LIMIT SWITCH INPUT BYTE
C280            TEACHING   EQU $C280   ;TEACHING_PNDT (REMOTE) INPUT BYTE
C200            IRQ_PORT   EQU $C200   ;IRQ INTERRUPT BYTE INPUT
C220            EXP_OUT_   EQU $C220   ;EXP_OUT_ADDR  EXPERIMENTAL BOARD OUTPUT BYTE
C2A0            EXP_IN_A   EQU $C2A0   ;EXP_IN_ADDR  EXPERIMENTAL BOARD INPUT BYTE
                ;
                ;PORT DEFINITIONS FOR HERO ROBOT ($C000-D000)
                ;
                ;Version 1.0, 1/16/84, written by A. H. Ballard
                ;
                ;
                ;(6801 ports are defined in HERODEF1.a)
```

```
                     ;Input ports:
                     ;
C003                 COL1     EQU $C003       ;Input from keypad column 1
C005                 COL2     EQU $C005       ;Input from keypad column 2
C006                 COL3     EQU $C006       ;Input from keypad column 3
                     ;
C220                 SNRPORT EQU $C220        ;Sonar range data (8 bits)
                     ;
C240                 SENSPORT EQU $C240       ;Light/sound intensity (8 bits)
                     ;
C260                 LIMPORT EQU $C260        ;Limit switches + miscellaneous
                     ;
                     ;  Bit 7 = Cassette tape input
                     ;  Bit 6 = Steering motor full CCW
                     ;  Bit 5 = Shoulder motor full DOWN
                     ;  Bit 4 = Head motor full CCW
                     ;  Bit 3 = Steering motor full CW
                     ;  Bit 2 = Shoulder motor full UP
                     ;  Bit 1 = Head motor full CW
                     ;  Bit 0 = Speech board ready for next phoneme
                     ;
C280                 REMPORT EQU $C280        ;Remote teach pendant, sleep sw.
                     ;
                     ;  Bit 7 = Select base (0) or arm (1) motion
                     ;
                     ;  Bits 6,5,4 = rotary selector
                     ;
                     ;    If base selected: 000 = stop base drive
                     ;                      001 = drive forward, slow
                     ;                      010 = drive forward, medium
                     ;                      011 = drive forward, fast
                     ;                      100 = (not used)
                     ;                      101 = drive reverse, slow
                     ;                      110 = drive reverse, medium
                     ;                      111 = drive reverse, fast
                     ;
                     ;    If arm selected:  000 = (not Used)
                     ;                      001 = select wrist pivot motor
                     ;                      010 = select wrist rotate motor
                     ;                      011 = select gripper motor
                     ;                      100 = (not used )
                     ;                      101 = select shoulder motor
                     ;                      110 = select arm extend motor
                     ;                      111 = select head motor
                     ;
                     ;  Bit 3 = 0 for left/down/in/open direction, 1 for OFF
                     ;
                     ;  Bit 2 = 0 for right/up/out/close direction, 1 for OFF
                     ;
                     ;  Bit 1 = Sleep switch (0 = sleep, 1 = normal)
                     ;
                     ;  Bit 0 = Pendant trigger (0 = released, 1 = active)
```

```
                          ;
C2A0                      EXPIPORT EQU $C2A0      ;Experiment board input port (8 bits)
                          ;
                          ;Input/output ports:
                          ;
C200                      IRQPORT EQU $C200       ;Interrupt request port
                          ;
                          ;  Input = read interrupt flags
                          ;  Output = reset interrupt flags
                          ;
                          ;    Bit 7 = experiment board interrupt
                          ;    Bit 6 = drive wheel spoke sensor
                          ;    Bit 5 = trigger on teach pendant
                          ;    Bit 4 = 1024 Hz clock
                          ;    Bit 3 = low logic battery sensor
                          ;    Bit 2 = low drive battery sensor
                          ;    Bit 1 = motion detector
                          ;    Bit 0 = sonar echo detector
                          ;
C300                      CLCKPORT EQU $C300      ;Set or read time/date
                          ;
                          ;  Bits 7-4 = Mode select
                          ;
                          ;   0100 = Write
                          ;   0101 = Hold
                          ;   1010 = Read
                          ;   (other codes not used)
                          ;
                          ;  Bits 3-0 = input/output data bus
                          ;
                          ;    Range = 0000-1001 (0-9) except:
                          ;
                          ;      For tens of hours, bit 2 = 0/1 for AM/PM
                          ;                         bit 3 = 0/1 for 12/24 hour clock
                          ;      For tens of days, bit 2 = 0/1 for normal/leap year
                          ;
                          ;  Clock digits are selected by bits 3-0 of ADRPORT
                          ;
                          ;
                          ;Output ports:
                          ;
C16F                      DG6ADD  EQU $C16F       ;Leftmost display digit
C15F                      DG5ADD  EQU $C15F
C14F                      DG4ADD  EQU $C14F
C13F                      DG3ADD  EQU $C13F
C12F                      DG2ADD  EQU $C12F
C11F                      DG1ADD  EQU $C11F       ;Rightmost display digit
                          ;
                          ;  See MHSYM681.A for definition of display symbols
                          ;
                          ;
C220                      EXPOPORT EQU $C220      ;Experiment board output port
```

```
                        ;
C240                    SPCHPORT EQU $C240      ;Speech port
                        ;
                        ;  Bits 7-6 = pitch select (00-11 for levels 1-4)
                        ;  Bits 5-0 = phoneme select (1 of 64)
                        ;
                        ;  See MHSYM681.A for definition of phoneme symbols
                        ;
C260                    STRPORT EQU $C260       ;Steer and wrist motor port
                        ;
                        ;  Bits 7-4 = output step pattern for steer motor
                        ;
                        ;  Bits 3-0 = output step pattern for:
                        ;
                        ;               Wrist pivot motor if ARMSEL = 01
                        ;               Wrist rotate motor if ARMSEL = 10
                        ;
                        ;  ARMSEL = bits 7,6 of ADRPORT
                        ;
C280                    EXTPORT EQU $C280     ;Extend/head/gripper/shoulder port
                        ;
                        ;  Bits 7-4 = output step pattern for:
                        ;
                        ;               Arm extend motor if ARMSEL = 01
                        ;               Head motor if ARMSEL = 10
                        ;
                        ;  Bits 3-0 = output step pattern for:
                        ;
                        ;               Gripper motor if ARMSEL = 01
                        ;               Shoulder motor if ARMSEL = 10
                        ;
                        ;  ARMSEL = bits 7,6 of ADRPORT
                        ;
C2A0                    DRVPORT EQU $C2A0       ;Base drive port
                        ;
                        ;  Bit 7 = direction (0 = forward, 1 = reverse)
                        ;
                        ;  Bits 6-0 = magnitude of speed
                        ;
                        ;    $00-3F = zero speed (dead zone)
                        ;    $40-FF = actual speed range
                        ;
C2C0                    ADRPORT EQU $C2C0       ;Address port
                        ;
                        ;  Bits 7-6 = ARMSEL
                        ;
                        ;    01 = select wrist pivot, arm extend, gripper motors
                        ;    10 = select wrist rotate, head, shoulder motors
                        ;    (other codes not used)
                        ;
                        ;  Bit 5 = speech strobe (for phoneme code)
                        ;
```

```
                ;  Bit 4 = (not used)
                ;
                ;  Bits 3-0 = select real-time clock digit
                ;
                ;    0000 = seconds (0-9)
                ;    0001 = tens of seconds (0-5)
                ;    0010 = minutes (0-9)
                ;    0011 = tens of minutes (0-5)
                ;    0100 = hours (0-9)
                ;    0101 = tens of hours (0-2)
                ;    0110 = day of week (0-6)
                ;    0111 = days (0-9)
                ;    1000 = tens of days (0-3)
                ;    1001 = months (0-9)
                ;    1010 = tens of months (0-1)
                ;    1011 = years (0-9)
                ;    1100 = tens of years (0-9)
                ;
C2E0            PWRPORT EQU $C2E0      ;Power control + miscellaneous
                ;
                ;  Bit 7 = Select light (1) or sound (0) sensor
                ;  Bit 6 = Main power (0/1 = OFF/ON)
                ;  Bit 5 = Light/sound sense board power
                ;  Bit 4 = Display board power
                ;  Bit 3 = Speech board power
                ;  Bit 2 = Motion detector power
                ;  Bit 1 = Sonar power
                ;  Bit 0 = Cassette tape output
                ;
                ;
                ;DEFINITION OF DISPLAY, PHONEME, AND OTHER SYMBOLS
                ;    USED WITH HERO ROBOT
                ;
                ;Version 1.0, 1/16/84, written by A. H. Ballard
                ;
                ;DISPLAY CODES
                ;
                ;      The HERO display has 6 digits.  Each digit is
                ;an LED display assembly having 7 segments and a dot
                ;(decimal point) arranged in the following pattern.
                ;
                ;       --                     6
                ;      !  !   SEGMENT        1   5    CODE BIT
                ;       --    PATTERN          0      ASSIGNMENT
                ;      !  !                  2   4
                ;       --.                    3  7
                ;
                ;The segments to be lit are identified by individual
                ;bits in an 8-bit code character, as shown.  Adding a
                ;dot after any character is achieved by adding 80 hex
                ;to the original display character code.
                ;
```

```
                        ;       A display character will be designated by
                        ;preceding it with the ` symbol (accent grave).
                        ;
                        ;HEX NUMBERS
                        ;
007E                    `       EQU $7E
0030                    `1      EQU $30
006D                    `2      EQU $6D
0079                    `3      EQU $79
0033                    `4      EQU $33
005B                    `5      EQU $5B
005F                    `6      EQU $5F
0070                    `7      EQU $70
007F                    `8      EQU $7F
007B                    `9      EQU $7B
0077                    `A      EQU $77
001F                    `B      EQU $1F
004E                    `C      EQU $4E
003D                    `D      EQU $3D
004F                    `E      EQU $4F
0047                    `F      EQU $47
                        ;
                        ;LETTERS
                        ;
0077                    `A      EQU $77         ;Same as hex A
007D                    `a      EQU $7D
001F                    `b      EQU $1F         ;Same as hex B
004E                    `C      EQU $4E         ;Same as hex C
000D                    `c      EQU $0D
003D                    `d      EQU $3D         ;Same as hex D
004F                    `E      EQU $4F         ;Same as hex E
0047                    `F      EQU $47         ;Same as hex F
005E                    `G      EQU $5E
0037                            EQU $37
0017                            EQU $17
0030                    i       EQU $30
003C                    `J      EQU $3C
000E                    `L      EQU $0E
0015                    `n      EQU $15
007E                    `O      EQU $7E         ;Same as hex 0
001D                    `o      EQU $1D
0067                    `p      EQU $67
0005                    `r      EQU $05
005B                    `S      EQU $5B
0070                    `t      EQU $70         ;Same as hex 7
003E                    `U      EQU $3E
001C                    `u      EQU $1C
003B                    `y      EQU $3B
                        ;
```

```
                        ;PUNCTUATION
                        ;
0001                    'hy     EQU $01         ;Hyphen, dash or minus
0008                    '_      EQU $08         ;Underscore
0080                    '.      EQU $80         ;Dot or period
00A0                    '!      EQU $A0         ;Exclamation point
0022                    'qt     EQU $22         ;Double quote
0065                    '?      EQU $65         ;Question mark
0000                    'sp     EQU $00         ;Space or blank
                        ;
                        ;PHONEME CODES
                        ;
                        ;       The speech synthesizer in the HERO accepts 8-bit
                        ;phoneme codes.  The lower 6 bits select 1 of 64 coded
                        ;phonemes, while the upper two bits select 1 of 4 pitch
                        ;levels.  The basic phoneme codes range from $00 to $3F
                        ;at pitch level 1.  To vary the pitch (inflection) to
                        ;get greater naturalness:
                        ;
                        ;           Add $40 to get pitch level 2
                        ;           Add $80 to get pitch level 3
                        ;           Add $C0 to get pitch level 4
                        ;
                        ;The STOP phoneme ($3F) is the only code that should
                        ;never be raised in pitch.  Raising it to $BF or $FF
                        ;would change its meaning.
                        ;
                        ;       A phoneme code will be designated by preceding
                        ;it with the ~ symbol (tilde).  Phoneme durations in
                        ;milliseconds are shown in parentheses in the comments.
                        ;
0000                    ~EH3    EQU $00         ;( 59) as in jack_e_t
0001                    ~EH2    EQU $01         ;( 71) as in e_nlist
0002                    ~EH1    EQU $02         ;(121) as in h_ea_vy
0003                    ~sp     EQU $03         ;( 47) short pause
0004                    ~DT     EQU $04         ;( 47) as in bu_tt_er
0005                    ~A2     EQU $05         ;( 71) as in m_a_de
0006                    ~A1     EQU $06         ;(103) as in m_a_de
0007                    ~ZH     EQU $07         ;( 90) as in a_z_ure
0008                    ~AH2    EQU $08         ;( 71) as in h_o_nest
0009                    ~I3     EQU $09         ;( 55) as in inhib_i_t
000A                    ~I2     EQU $0A         ;( 80) as in i_nhibit
000B                    ~I1     EQU $0B         ;(121) as in inh_i_bit
000C                    ~M      EQU $0C         ;(103) as in m_at
000D                    ~N      EQU $0D         ;( 80) as in su_n
000E                    ~B      EQU $0E         ;( 71) as in b_ag
000F                    ~V      EQU $0F         ;( 71) as in v_an
0010                    ~CH     EQU $10         ;( 71) as in ch_ip (~T first)
0011                    ~SH     EQU $11         ;(121) as in sh_op
0012                    ~Z      EQU $12         ;( 71) as in z_oo
0013                    ~AW1    EQU $13         ;(146) as in l_aw_ful
0014                    ~NG     EQU $14         ;(121) as in thi_ng
```

```
0015          ~AH1   EQU $15      ;(146) as in f_a_ther
0016          ~OO1   EQU $16      ;(103) as in l_oo_king
0017          ~OO    EQU $17      ;(185) as in b_oo_k
0018          ~L     EQU $18      ;(103) as in l_and
0019          ~K     EQU $19      ;( 80) as in tric_k
001A          ~J     EQU $1A      ;( 47) as in ju_dg_e (~D first)
001B          ~H     EQU $1B      ;( 71) as in h_ello
001C          ~G     EQU $1C      ;( 71) as in g_et
001D          ~F     EQU $1D      ;(103) as in f_ast
001E          ~D     EQU $1E      ;( 55) as in pai_d
001F          ~S     EQU $1F      ;( 90) as in pa_ss
0020          ~A     EQU $20      ;(185) as in d_ay
0021          ~AY    EQU $21      ;( 65) as in d_ay
0022          ~Y1    EQU $22      ;( 80) as in y_ard
0023          ~UH3   EQU $23      ;( 47) as in miss_io_n
0024          ~AH    EQU $24      ;(250) as in m_o_p
0025          ~P     EQU $25      ;(103) as in p_ast
0026          ~O     EQU $26      ;(185) as in c_o_ld
0027          ~I     EQU $27      ;(185) as in p_i_n
0028          ~U     EQU $28      ;(185) as in m_o_ve
0029          ~Y     EQU $29      ;(103) as in an_y
002A          ~T     EQU $2A      ;( 71) as in t_ap
002B          ~R     EQU $2B      ;( 90) as in r_ed
002C          ~E     EQU $2C      ;(185) as in m_ee_t
002D          ~W     EQU $2D      ;( 80) as in w_in
002E          ~AE    EQU $2E      ;(185) as in d_a_d
002F          ~AE1   EQU $2F      ;(103) as in a_fter
0030          ~AW2   EQU $30      ;( 90) as in s_a_lty
0031          ~UH2   EQU $31      ;( 71) as in a_bout
0032          ~UH1   EQU $32      ;(103) as in u_ncle
0033          ~UH    EQU $33      ;(185) as in c_u_p
0034          ~O2    EQU $34      ;( 80) as in f_o_r
0035          ~O1    EQU $35      ;(121) as in ab_oa_rd
0036          ~IU    EQU $36      ;( 59) as in y_ou
0037          ~U1    EQU $37      ;( 90) as in y_ou
0038          ~THV   EQU $38      ;( 80) as in th_e
0039          ~TH    EQU $39      ;( 71) as in th_in
003A          ~ER    EQU $3A      ;(146) as in b_ir_d
003B          ~EH    EQU $3B      ;(185) as in g_e_t
003C          ~E1    EQU $3C      ;(121) as in b_e
003D          ~AW    EQU $3D      ;(250) as in c_a_ll
003E          ~lp    EQU $3E      ;(185) long pause
003F          ~stop  EQU $3F      ;( 47) no sound
              ;
2143                 ORG    02143H
              ;
              ;
```

```
                ;HERO COMMAND INTERFACE ROUTINES - C Language Interface
                ;
                ;Version 6
                ;Written by A. H. Ballard, 11/21/83
                ;Updated 7-6-84
                ;
                ;   These routines allow external programs to interface
                ;with the motors, sensors, speech, and other subsystems
                ;in the Hero 1.x robot.  The interface is implemented
                ;as a series of subroutine calls.  The parameters, or
                ;operands, for each subroutine are first pushed onto the
                ;stack.  All operands have 16 bits, with right justifi-
                ;cation.  The last operand is stacked first, and the
                ;first operand is stacked last.  A subroutine call is
                ;then made to one of the 85 jump vectors provided at the
                ;beginning of this program.
                ;
                ;   In this version, 35 of the 38 interpreted HERO
                ;commands are included in the jump table, althougth the
                ;SLEEP is not actually implemented in this version.  The
                ;CRL (change to robot language) and CML (change to
                ;machine language) are not included because they are
                ;not appropiate for external use.  The JISP (jump if
                ;speaking) command is actually a subroutine call to a
                ;lower level in the phoneme tree, and is not included
                ;either.  Room for expansion to a total of 85 commands
                ;is provided in the jump table.
                ;
                ;   Maximum use is made of subroutines which already
                ;exist in the Hero ROM.  Where no operands need to be
                ;passed, it is usually possible to jump directly to an
                ;existing ROM subroutine.  Where one or more operands
                ;have to be passed, they are reformatted in this pro-
                ;gram to satisfy the entry conditions required by
                ;existing ROM subroutines.
                ;
                ;Entry points in existing ROM:
                ;
E390            ABM     EQU $E390       ;First of 16 direct entry points
E399            ASM     EQU $E399       ;  (see jump table for comments)
E39D            AAM     EQU $E39D
E3B2            ASP     EQU $E3B2
E417            ZERO    EQU $E417
F423            RTE     EQU $F423
E0A9            ENEYE   EQU $E0A9
E0AD            ENEAR   EQU $E0AD
E0B8            ENSON   EQU $E0B8
E0BC            ENMDT   EQU $E0BC
E0C0            ENDIS   EQU $E0C0
E0CA            DISEYE  EQU $E0CA
E0D5            DISEAR  EQU $E0D5
E0E0            DISSON  EQU $E0E0
```

```
E0E4                    DISMDT  EQU $E0E4
E0E8                    DISDIS  EQU $E0E8      ;Last of 16 direct entry points
                        ;
E35F                    SPKEXTRN EQU $E35F     ;External entry for speaking
E499                    PSE.EXT EQU $E499      ;External entry for PAUSE
E169                    MVR.EXT EQU $E169      ;Ext. entry for move relative
E114                    MVI.EXT EQU $E114      ;Ext. entry for move immediate
E151                    MVX.EXT EQU $E151      ;Ext. entry for move indexed
E12A                    MVE.EXT EQU $E12A      ;Ext. entry for move extended
E1FC                    MVALLX  EQU $E1FC      ;External entry for move all
F65B                    CLRDIS  EQU $F65B      ;Clear HERO Display
F7E5                    OUTSTR  EQU $F7E5      ;Output string to HERO Display
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;Exit points in existing ROM:
                        ;
                        ;These locations are jumps or jsr's to USER routines located at the
                        ; addresses stated:
                        ;
                        ;               MOD-HERO INTERPRETER MODE
                        ;THE FOLLOWING EXPECT A JUMP OR RTS INSTRUCTION AT THE TO ADDR
                        ;
E001                    UEXCYCLE  EQU   $E001  ;JSR TO USRCYCLE---START OF INTERPRETER CYCLE
E024                    UEXINTRP  EQU   $E024  ;JSR TO USRINTRP---EXTENSION OF INTERPRETER CODES
                        ;
                        ;               IRQ INTERRUPT HANDLERS
                        ;THE FOLLOWING EXPECT A JUMP OR RTS INSTRUCTION AT THE TO ADDR
                        ;
EFCC                    UEXIRQ    EQU   $EFCC  ;JSR TO USRIRQ---CUSTOMIZED IRQ HANDLER
F01E                    UEXCLOCK  EQU   $F01E  ;JSR TO USRCLOCK---CUSTOMIZED CLOCK INT HANDLER
F02E                    UEXSPD  EQU     $F02E  ;JSR TO USRSPD---CUSTOMIZED DRIVE SPEED HANDLER
F256                    UEXLLB  EQU     $F256  ;JSR TO USRLLB---LOW LOGIC BATTERY VOLTAGE
F2D9                    UEXLDB  EQU     $F2D9  ;JSR TO USRLDB---LOW DRIVE BATTERY VOLTAGEE
F004                    UEXMDT  EQU     $F004  ;JSR TO USRMDT---MOTION DETECTOR HANDLER
F00B                    UEXTRG  EQU     $F00B  ;JSR TO USRTRG---TEACHING PENDENT TRIGGER
F012                    UEXEXP  EQU     $F012  ;JSR TO USREXP---EXPERIMENTAL PORT IRQ INT
                        ;
                        ;               MOD-HERO EXECUTIVE MODE
                        ;
F53B                    UEXUSER1  EQU   $F53B  ;JSR TO USER1---USER1 KEY DEPRESSED
                        ;                      ; LOADS LOCATION USER2 INTO A,B REG BEFORE JUMP
F541                    UEXUSER2  EQU   $F541  ;JSR TO USER2---USER2 KEY DEPRESSED
                        ;                      ; LOADS LOCATION USER2 INTO A,B REG BEFORE JUMP
F547                    UEXUSER3  EQU   $F547  ;JSR TO USER3---USER3 KEY DEPRESSED
                        ;                      ; LOADS LOCATION USER3 INTO A,B REG BEFORE JUMP
                        ;
                        ;
                        ;
                        ;
```

```
                   ; USER DECLARED TABLES
0023               NVEC    EQU 35          ;Current number of jump vectors
                   ;
                   ;
                   ;
2143               SPKTAB  RMB 1   ;Table of phrase addresses
                   ;
2144               POSITAB RMB 1   ;Table of motor positions
                   ;
                   ;
                   ;
                   ;RAM locations for parameters:
                   ;
0020               CPOSTAB EQU $0020       ;CURRENT MOTOR POSITION TABLE
002B               ACTSTAT EQU $002B       ;Activity status byte
0F20               ACTMASK EQU $0F20       ;Mask to select ACTSTAT bits
                   ;
                   ;
C220               EXP_OUT_    EQU  $C220  ;EXP_OUT_ADDR  EXPERIMENTAL BOARD OUTPUT BYTE
C2A0               EXP_IN_A    EQU  $C2A0  ;EXP_IN_ADDR  EXPERIMENTAL BOARD INPUT BYTE
                   ;
0023               NVEC    EQU 35          ;Current number of jump vectors
                   ;
                   ;
                   ;
                   ;
2145    39         RTRN    RTS             ;Default for unimplemented
                                           ;routines
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;activity()
                   activity
                   ;{
                   ;       char *stat;
                   ; /* check if motors are currently running, return true if they are,
                   ;       false otherwise.  */
                   ;
                   ;       stat = ACTSTAT;
                   ;       return (*stat & 0x0F);
2146    C6 0F              LDAB    #$0F    ;LOAD MASK FOR ACTSTAT
2148    20 0E              BRA BUSY
                   ;}
                   ;
214A    C6 01      BBSY    LDAB #01        ;Activity bit for base
214C    20 0A              BRA BUSY
214E    C6 02      SBSY    LDAB #02        ;Activity bit for steering
2150    20 06              BRA BUSY
2152    C6 04      ABSY    LDAB #04        ;Activity bit for arm/head
2154    20 02              BRA BUSY
```

```
2156   C6 08       VBSY     LDAB #08      ;Activity bit for voice (speech)
                   ;
2158   D4 2B       BUSY     ANDD ACTSTAT  ;Test busy status
215A   4F                   CLRA
215B   39                   RTS           ;Return with D = 0 if not busy
                   ;
                   ;
                   ;
                   ;
                   ;
215C   86 08       SPWX     LDAA #08      ;Setup wait for speech
215E   20 01                BRA SPCX.1
                   ;
2160   4F          SPCX     CLRA          ;Setup to continue
2161   B7 0F 20    SPCX.1   STAA ACTMASK
2164   30                   TSX
2165   E6 03                LDAB 3,X      ;B = phrase number
2167   58                   ASLB          ;B = address offset
2168   FE 21 43             LDX SPKTAB    ;Point to phrase table
216B   3A                   ABX           ;Point to phrase address
216C   E6 00                LDAB 0,X
216E   A6 01                LDAA 1,X      ;BA = phrase address
2170   7E E3 5F             JMP SPKEXTRN
                   ;
2173   86 08       SPWE     LDAA #08      ;Setup wait for speech
2175   20 01                BRA SPCE.1
                   ;
2177   4F          SPCE     CLRA          ;Setup to continue
2178   B7 0F 20    SPCE.1   STAA ACTMASK
217B   30                   TSX
217C   E6 02                LDAB 2,X
217E   A6 03                LDAA 3,X      ;BA = phrase address
2180   7E E3 5F             JMP SPKEXTRN
                   ;
2183   30          PAUSE    TSX
2184   EE 02                LDX 2,X       ;IX = count (1/16 sec steps)
2186   7E E4 99             JMP PSE.EXT
                   ;
2189   7E 21 45    SLEEP    JMP RTRN      ;Not currently implemented
                   ;
218C   86 07       MVWR     LDAA #07      ;Setup motor wait bits
218E   20 01                BRA MVCR.1
                   ;
2190   4F          MVCR     CLRA          ;Setup to continue
2191   B7 0F 20    MVCR.1   STAA ACTMASK
2194   30                   TSX
2195   EE 02                LDX 2,X       ;IX = 2-byte operand
2197   7E E1 69             JMP MVR.EXT   ;Use existing ROM routine
                   ;
219A   86 07       MVWI     LDAA #07      ;Setup motor wait bits
219C   20 01                BRA MVCI.1
                   ;
```

```
219E  4F          MVC1    CLRA            ;Setup to continue
219F  B7 0F 20    MVC1.1  STAA ACTMASK
21A2  30                  TSX
21A3  EE 02               LDX 2,X         ;IX = 2-byte operand
21A5  7E E1 14            JMP MV1.EXT     ;Use existing ROM routine
                  ;
21A8  86 07       MVWX    LDAA #07        ;Setup motor wait bits
21AA  20 01               BRA MVCX.1
                  ;
21AC  4F          MVCX    CLRA            ;Setup to continue
21AD  B7 0F 20    MVCX.1  STAA ACTMASK
21B0  30                  TSX
21B1  E6 03               LDAB 3,X        ;B = position index
21B3  58                  ASLB            ;B = address offset
21B4  FE 21 44            LDX POSITAB     ;Point to table of positions
21B7  3A                  ABX             ;Point to selected position
21B8  7E E1 51            JMP MVX.EXT     ;Use existing ROM routine
                  ;
21BB  86 07       MVWE    LDAA #07        ;Setup motor wait bits
21BD  20 01               BRA MVCE.1
                  ;
21BF  4F          MVCE    CLRA            ;Setup to continue
21C0  B7 0F 20    MVCE.1  STAA ACTMASK
21C3  30                  TSX
21C4  EE 02               LDX 2,X         ;IX = address of operand
21C6  7E E1 2A            JMP MVE.EXT     ;Use existing ROM routine
                  ;
21C9  30          MVALL   TSX
21CA  08                  INX
21CB  08                  INX             ;IX will point to 1st operand
21CC  A6 01               LDAA 1,X
21CE  A7 00               STAA 0,X        ;1st operand = EXTPOS
21D0  A6 03               LDAA 3,X
21D2  A7 01               STAA 1,X        ;2nd operand = SHLPOS
21D4  A6 05               LDAA 5,X
21D6  A7 02               STAA 2,X        ;3rd operand = ROTPOS
21D8  A6 07               LDAA 7,X
21DA  A7 03               STAA 3,X        ;4th operand = PVTPOS
21DC  A6 09               LDAA 9,X
21DE  A7 04               STAA 4,X        ;5th operand = GRPPOS
21E0  A6 0B               LDAA 11,X
21E2  A7 05               STAA 5,X        ;6th operand = HEDPOS
21E4  A6 0D               LDAA 13,X
21E6  A7 06               STAA 6,X        ;7th operand = STRPOS
21E8  7E E1 FC            JMP MVALLX
                  ;
21EB  30          MMV     TSX
21EC  C6 40               LDAB    #01000000B      ;SET FOR SLOW SPEED
21EE  A6 03               LDAA    3,X             ;A NOW IS MOTOR INDEX
21F0  04                  LSRD                    ;ALIGN BITS IN B REG
21F1  04                  LSRD
21F2  04                  LSRD
```

```
21F3   A6 05              LDAA   5,X           ;A IS POSITION
21F5   36                 PSHA             ; NOTE FOLLOWING REVERSAL FROM STANDARD PUSH
21F6   37                 PSHB             ; <<<<<<<------------------- ORDER
21F7   38                 PULX                 ; IX = 2-BYTE OPERAND
21F8   7E E1 14           JMP    MVI.EXT       ; GO TO HERO ROM AND RETURN
                   ;
                   ;
0F24               SRC.H  EQU    $0F24
0F22               DEST.H EQU    $0F22
E291               MOVE.H EQU    $E291
                   ;
21FB   CE 22 0C    MAXPOS LDX    #MAXTBL       ;POINT TO TABLE OF MAX POSITIONS
21FE   FF 0F 24           STX    SRC.H         ;$0F24 IN HERO RAM
2201   CE 00 20           LDX    #CPOSTAB      ;$0020 IN HERO RAM
2204   FF 0F 22           STX    DEST.H        ; $0F22 IN HERO RAM
2207   C6 07              LDAB   #7            ; COUNT IS 7 BYTES
2209   7E E2 91           JMP    MOVE.H        ;$E291  IN HERO ROM
220C   98 86 93 A5 MAXTBL FCB    $98,$86,$93,$A5,$75,$C9,$93
2210   75 C9 93
                   ;
                   ;
                   ;
                   ;EXPOUT(c)
                   ;      char c;
                   EXPOUT
                   ;{     EXPOUTPUT = c; /* output the data to the experimenter port */
2213   30                 TSX
2214   E6 03              ldab   3,X
2216   F7 C2 20           stab   $C220
                   ;}
2219   39                 RTS
                   ;EXPINP()
                   EXPINP
                   ;{     return EXPINPUT ; /* input the data from the experimenter port */
221A   4F                 clra
221B   F6 C2 A0           ldab   $C2A0
221E   39                 RTS
                   ;
                   ;
                   ;  The following equates state how the functions in Menos utilize the
                   ;routines above.  Please do not assume that Menos uses all of the
                   ;above routines.  Menos must never wait too long for a routine (such
                   ;as zero) to execute, and thus has equivalent routines, with the names
                   ;given as below, which perform the function of the above routines.
                   ;
                   ;
E390               abmvec EQU   ABM       ;Abort base motor
E399               asmvec EQU   ASM       ;Abort steering motor
E39D               aamvec EQU   AAM       ;Abort arm/head motors
E3B2               aspvec EQU   ASP       ;Abort speech
214A               bbsyvec EQU  BBSY      ;Test for base busy
214E               sbsyvec EQU  SBSY      ;Test for steering busy
```

```
2152    absyvec EQU   ABSY     ;Test for arm/head busy
2156    vbsyvec EQU   VBSY     ;Test for voice (speech) busy
E417    zrovec  EQU   ZERO     ;Initialize all stepper motors
F423    rtevec  EQU   RTE      ;Return to executive mode
E0A9    eldvec  EQU   ENEYE    ;Enable light detector (eye)
E0AD    esdvec  EQU   ENEAR    ;Enable sound detector (ear)
E0B8    eurvec  EQU   ENSON    ;Enable ultrasonic ranger
E0BC    emdvec  EQU   ENMDT    ;Enable motion detector
E0C0    edsvec  EQU   ENDIS    ;Enable display
E0CA    dldvec  EQU   DISEYE   ;Disable light detector (eye)
E0D5    dsdvec  EQU   DISEAR   ;Disable sound detector (ear)
E0E0    durvec  EQU   DISSON   ;Disable ultrasonic ranger
E0E4    dmdvec  EQU   DISMDT   ;Disable motion detector
E0E8    ddsvec  EQU   DISDIS   ;Disable display
215C    spwxvec EQU   SPWX     ;Speak and wait (indexed)
2160    spcxvec EQU   SPCX     ;Speak and continue (indexed)
2173    spwevec EQU   SPWE     ;Speak and wait (extended)
2177    spcevec EQU   SPCE     ;Speak and continue (extended)
2183    psevec  EQU   PAUSE    ;Pause for specified time
2189    slpvec  EQU   SLEEP    ;Sleep for specified time
219A    mwivec  EQU   MVWI     ;Move and wait (immediate)
219E    mcivec  EQU   MVCI     ;Move and continue (immediate)
218C    mwrvec  EQU   MVWR     ;Move relative, wait (immediate)
2190    mcrvec  EQU   MVCR     ;Move relative, continue (immed)
21A8    mwxvec  EQU   MVWX     ;Move and wait (indexed)
21AC    mcxvec  EQU   MVCX     ;Move and continue (indexed)
21B8    mwevec  EQU   MVWE     ;Move and wait (extended)
21BF    mcevec  EQU   MVCE     ;Move and continue (extended)
21C9    mvlvec  EQU   MVALL    ;Move all 7 stepper motors
21FB    smtrmax EQU   MAXPOS   ;Set all motor position registers to max
21EB    mtrmvec EQU   MMV              ;Special MVCI at medium speed
221A    expinpve EQU  EXPINP   ;Get experimental board input
2213    expoutve EQU  EXPOUT   ;Output byte to experimental port
        ;
        ;
        ;
F37E    PWRON   EQU    $F37E   ;WARM START ADDRESS---NO RETURN EXPECTED
7EFD    RTMH    EQU    $7EFD   ;RETURN TO MOD-HERO IN VERSION 1.1B OF VAMP
221F            END

No error(s).
```

SYMBOL TABLE FOR FILE: DRIVEDEF.A

AAM	E39D	ABM	E390	ABSY	2152	ABTSTAT	002C
ACNT	0015	ACTIVATE	0F21	ACTMASK	0F20	ACTSTAT	002B
ADDR	0F2E	ADDRPTR	0F2C	ADRPCPY	0F02	ADRPORT	C2C0
ARMBACK	0F41	ARMON	0F40	ARMSTOP	0F42	ARMSTP	0EF0
ASM	E399	ASP	E382	BASEOP	0F46	BBSY	214A
BKPNTSTK	0FEC	BKTBL	0FE0	BUSY	2158	CLCKPORT	C300
CLCKTIME	0EFC	CLRDIS	F65B	CNTR	0009	COL1	C003
COL2	C005	COL3	C006	CPOSTAB	0020	CR3	000F
DDR1	0000	DDR1CPY	0006	DDR2	0001	DDR3	0004
DDR4	0005	DELAYTM	0F43	DEST	0F22	DEST.H	0F22
DG1ADD	C11F	DG2ADD	C12F	DG3ADD	C13F	DG4ADD	C14F
DG5ADD	C15F	DG6ADD	C16F	DIGADD	0FF2	DIR	0F3C
DISDIS	E0E8	DISEAR	E0D5	DISEYE	E0CA	DISLMDE	0F4E
DISMOT	E0E4	DISSON	E0E0	DR1	0002	DR1CPY	000F
DR2	0003	DR2CPY	0004	DR3	0006	DR4	0007
DRVNO	0EFD	DRVCOM	0029	DRVON	0F3E	DRVPCPY	0F00
DRVPNEW	0F01	DRVPORT	C2A0	DRVPOS	0027	DSHDIR	0FFB
DSHPTR	0FF9	ENDIS	E0C0	ENEAR	E0AD	ENEYE	E0A9
ENMOT	E0BC	ENSON	E0B8	EXPINP	221A	EXPIPORT	C2A0
EXPOPORT	C220	EXPOUT	2213	EXP_IN_A	C2A0	EXP_OUT_	C220
EXTPORT	C280	EXTPOS	0020	FRSTADDR	0F32	GRPPOS	0024
HEDPOS	0025	HUNGRY	002F	ICR1	000D	ICR2	001E
IRQPCPY	0F04	IRQPORT	C200	IRQ_PORT	C200	ISARM	0F3D
ISOLD	0F3A	LASTADDR	0F34	LIMIT	C260	LIMPORT	C260
LRNMODE	0F36	MAXPOS	21FB	MAXTBL	220C	MCB	0F1E
MMV	21EB	MODE	0EE1	MONSTK	0F98	MOVE.H	E291
MTR	0F4A	MTRSPD	0F1B	MVALL	21C9	MVALLOP	0F26
MVALLX	E1FC	MVCE	21BF	MVCE.1	21C0	MVCI	219E
MVCI.1	219F	MVCR	2190	MVCR.1	2191	MVCX	21AC
MVCX.1	21AD	MVE.EXT	E12A	MVI.EXT	E114	MVR.EXT	E169
MVWE	21BB	MVWI	219A	MVWR	218C	MVWX	21A8
MVX.EXT	E151	NVEC	0023	OCR1	000B	OCR2	001A
OCR3	001C	OLDADDR	0F30	OLDDIR	0F38	OLDDIS	0F4C
OLDLMDE	0F4F	OLDPOS	0F39	OLDSLCT	0F37	OPCODE	0F47
OUTSTR	F7E5	PAUSE	2183	PCSAVE	0EE2	POS	0F48
POSITAB	2144	PSE.EXT	E499	PVTPOS	0023	PWRON	F37E
PWRPCPY	0F03	PWRPORT	C2E0	PWRSTK	0F18	PWRTEMP	0EDF
RAMC	0014	RDR	0012	REMPORT	C280	REVADDR	0F2A
REVPOS	0F28	RMCR	0010	ROTPOS	0022	RTE	F423
RTMH	7EFD	RTRN	2145	RXBUFF	0EDE	RXSTAT	0EDD
SBSY	214E	SENSE	C240	SENSPORT	C240	SHLPOS	0021
SLCT	0F3B	SLEEP	2189	SLEEPER	002D	SNRHIT	0030
SNRPORT	C220	SNRRNG	0031	SNRTIMR	C220	SPCE	2177
SPCE.1	2178	SPCHPORT	C240	SPCX	2160	SPCX.1	2161
SPD	0F4B	SPKEXTRN	E35F	SPKTAB	2143	SPSAVE	0F56
SPSKAREA	0F05	SPSKFWA	0F15	SPSKLWA	0F17	SPSKPTR	0F19
SPWE	2173	SPWX	215C	SRC	0F24	SRC.H	0F24
STPPTR	0F52	STRCTADR	0F1C	STRGON	0F3F	STRPCPY	0EFF
STRPORT	C260	STRPOS	0026	STRSTP	0EE4	SYSSWI	0FF6

T0	0059	T1	005B	T2	005D	TCR1	0017
TCR2	0018	TCSR	0008	TDR	0013	TE0	0FEE
TE1	0FF0	TEACHING	C200	TEMP1	0F54	TIME1	0FFC
TIME2	0FFE	TRCS	0011	TRIGIN	0F45	TSR	0019
UEXCLOCK	F01E	UEXCYCLE	E001	UEXEXP	F012	UEXINTRP	E024
UEXIRQ	EFCC	UEXLDB	F209	UEXLLB	F256	UEXMDT	F004
UEXSPD	F02E	UEXTRG	F00B	UEXUSER1	F53B	UEXUSER2	F541
UEXUSER3	F547	USER1	0050	USER2	0053	USER3	0056
USERS	0FF4	USRCLOCK	003B	USRCYCLE	0032	USREXP	004D
USRINTRP	0035	USRIRQ	0038	USRLDB	0044	USRLLB	004!
USRMDT	0047	USRRAM	005F	USRSPD	003E	USRSTK	0F58
USRTRG	004A	VBSY	2156	XTEMP	0F50	ZERO	E417
`!	00A0	`.	0080	`0	007E	`1	0030
`2	006D	`3	0079	`4	0033	`5	005B
`6	005F	`7	0070	`8	007F	`9	007B
`?	0065	`A	0077	`B	001F	`C	004E
`D	0030	`E	004F	`F	0047	`G	005E
`H	0037	`I	0030	`J	003C	`L	000E
`O	007E	`S	005B	`U	003E	`_	0008
`a	007D	`b	001F	`c	000D	`d	003D
`h	0017	`hy	0001	`n	0015	`o	001D
`p	0067	`qt	0022	`r	0005	`sp	0000
`t	0070	`u	001C	`y	003B	aamvec	E39D
abmvec	E390	absyvec	2152	activity	2146	asmvec	E399
aspvec	E3B2	bbsyvec	214A	ddsvec	E0E8	dldvec	E0CA
dmdvec	E0E4	dsdvec	E0D5	durvec	E0E0	edsvec	E0C0
eldvec	E0A9	emdvec	E0BC	esdvec	E0AD	eurvec	E0B8
expinpve	221A	expoutve	2213	mcevec	21BF	mcivec	219E
mcrvec	2190	mcxvec	21AC	mpos	0020	mtrmvec	21EB
mvlvec	21C9	mwevec	21BB	mwivec	219A	mwrvec	218C
mwxvec	21A8	psevec	2183	rtevec	F423	sbsyvec	214E
slpvec	2189	smtrmax	21FB	spcevec	2177	spcxvec	2160
spwevec	2173	spwxvec	215C	vbsyvec	2156	zrcvec	E417
~A	0020	~A1	0006	~A2	0005	~AE	002E
~AE1	002F	~AH	0024	~AH1	0015	~AH2	0008
~AW	0030	~AW1	0013	~AW2	0030	~AY	0021
~B	000E	~CH	0010	~D	001E	~DT	0004
~E	002C	~E1	003C	~EH	003B	~EH1	0002
~EH2	0001	~EH3	0000	~ER	003A	~F	001D
~G	001C	~H	001B	~I	0027	~I1	000B
~I2	000A	~I3	0009	~IU	0036	~J	001A
~K	0019	~L	0018	~M	000C	~N	000D
~NG	0014	~O	0026	~O1	0035	~O2	0034
~OO	0017	~OO1	0016	~P	0025	~R	002B
~S	001F	~SH	0011	~T	002A	~TH	0039
~THV	0038	~U	0028	~U1	0037	~UH	0033
~UH1	0032	~UH2	0031	~UH3	0023	~V	000F
~W	002D	~Y	0029	~Y1	0022	~Z	0012
~ZH	0007	~lp	003E	~sp	0003	~stop	003F

APPENDIX F

Navigation Computer Parts List

U#	LOC.	TYPE	#PINS	PARTS USED	+5VDC	GROUND	+6VDC	+12VDC	-12VDC
1		DC-DC	24	ALL	1	10,12,13,15		11,14	
0	AA37	DC-DC	24	ALL	1	11,12,13,14			10,15
1	AA15	6802	40	ALL	2,3,8,35	1,21			
2	AD1	74LS244	20	ALL	20	10			
3	AD14	74LS244	20	ALL	20	10			
4	AD26	74LS245	20	ALL	20	10			
5	AD38	74LS245	20	ALL	20	10			
6	AF1	74LS04	14	1,2,3,4	14	7			
7	AF10	74LS14	14	1,2,3,4,11,15, 13,14	14	7			
8	AF18	74LS138	16	ALL	16	8			
9	AF28	74LS138	16	ALL	16	8			
10	AF38	74LS00	14	1,2,3,4,5,6	14	7			
11	AH1	EDH8808	28	ALL	2[illegible]	14			
12	AH18	EDH8808	28	ALL	28	14			
13	AH34	EDH8808	28	ALL	28	14			
14	BB1	EDH8808	28	ALL	28	14			
15	BB18	EDH8808	28	ALL	28	14			
16	BB34	EDH8808	28	ALL	28	14			
17	BE1	6850	24	ALL	12	1			
18	BE18	2764	28	ALL	1,26,28	14			
19	BE34	14411	24	ALL	2[illegible]	12			
20	BH1	6850	24	ALL	12	1			
21	BH20	6821	40	ALL	20	1			
22	BH42	1488	14			7		14	1
23	CB1	74125	14	1,2,3,4,5,6	14	1,7			
24	CB20	6821	40	ALL	20	1			
25	CB42	1[illegible]89	14		14	7			
26	CE1	6840	28	T1,T2	14	1			
27	CE18	6840	28	T1,T2	14	1			
28	CE34	6840	28	T3	14	1			
29	CJ1	75468	16	ALL	9	8			
30	CJ10	75468	16	ALL	9	8			
31	CJ20	75468	16	ALL	9	8			
32	CJ30	75468	16	ALL	9	8			
33	CJ4[illegible]	75468	16	ALL	9	8			
34	DC1	RELAY	14	A0			2		
35	DC9	RELAY	14	B0			2		
36	DC17	RELAY	14	C0			2		
37	DC25	RELAY	14	D0			2		

U#	LOC.	TYPE	#PINS	PARTS USED	+5VDC	GROUND	+6VDC	+12VDC	-12VDC
38	DC33	S2	16	ALL					
39	DC42	S1	16	ALL					
40	DE1	RELAY	14	A1			2		
41	DE9	RELAY	14	B1			2		
42	DE17	RELAY	14	C1			2		
43	DE25	RELAY	14	D1			2		
44	DE33	74LS74	14		14	7			
45	DE42	74LS74	14		14	7			
46	DG1	RELAY	14	A2			2		
47	DG9	RELAY	14	B2			2		
48	DG17	RELAY	14	C2			2		
49	DG25	RELAY	14	D2			2		
50	DG33	75468	16	SPARE	9	8			
51	DG42	7404	14	1,2,3,4,5,6,8,9	14	7			
52	DJ1	RELAY	14	A4			2		
53	DJ9	RELAY	14	B4			2		
54	DJ17	RELAY	14	C4			2		
55	DJ25	RELAY	14	D4			2		
56	DJ33	7404	14	1,2,3,4,5,6,8,9	14	7			
57	EC1	RELAY	14	A5			2		
58	EC9	RELAY	14	B5			2		
59	EC17	RELAY	14	C5			2		
60	EC25	RELAY	14	D5			2		
61	EE1	RELAY	14	A6			2		
62	EE9	RELAY	14	B6			2		
63	EE17	RELAY	14	C6			2		
64	EE25	RELAY	14	D6			2		
65	EG1	RELAY	14	SPARE			2		
66	EG9	RELAY	14	SPARE			2		
67	EG17	RELAY	14	SPARE			2		
68	EG25	RELAY	14	SPARE			2		
69	EJ1	RELAY	14	SPARE			2		
70	EJ9	RELAY	14	SPARE			2		
71	EJ17	RELAY	14	SPARE			2		
72	EJ25	RELAY	14	SPARE			2		
73	EB34	2764	28	ALL	1,26,28	14			
74	CB10	74LS138	16		16	8			
75	EE34	6850	24		12	1			
76	FB33	74LS138	16	ALL	16	8			
77	FB43	74LS138	16	ALL	16	8			
78	OSE	6840	28	ALL	19	20			
79	OSE	6840	28	ALL	19	20			
80	OSE	6840	28	ALL	19	20			
81	OSE	CONNECTOR							
82	OSE	CONNECTOR							
83	EJ43	74LS123	16		16	8			
84	OSE	DM9602	16	ALL	16	8			
85	OSE	DM9602	16	ALL	16	8			
86	OSE	DM9602	16	ALL	16	8			

U# LOC.	TYPE	#PINS	PARTS USED	+5VDC	GROUND	+6VDC	+12VDC	-12VDC
CD1	420 KHz Clock Board							
EB1	TIP125 and D1-D9 (silicon signal diodes)							
	Sonar Boards (A,B,C,D)							
	S1 (Reset, momentary)							
DJ41	RESISTOR PACK (5.1 K)							
AA13	6 MHz CRYSTAL							
BF47	1.8432 MHz CRYSTAL							
	RESISTOR PACK (5.1 K) (BUTTOM OF NAV CPU BOARD)							
	24 POLAROID SONAR TRANSDUCERS (ON ROBOT BODY PANNELS)							

APPENDIX G

NAVIGATION COMPUTER SCHEMATICS

Appendix G contains the following schematics:

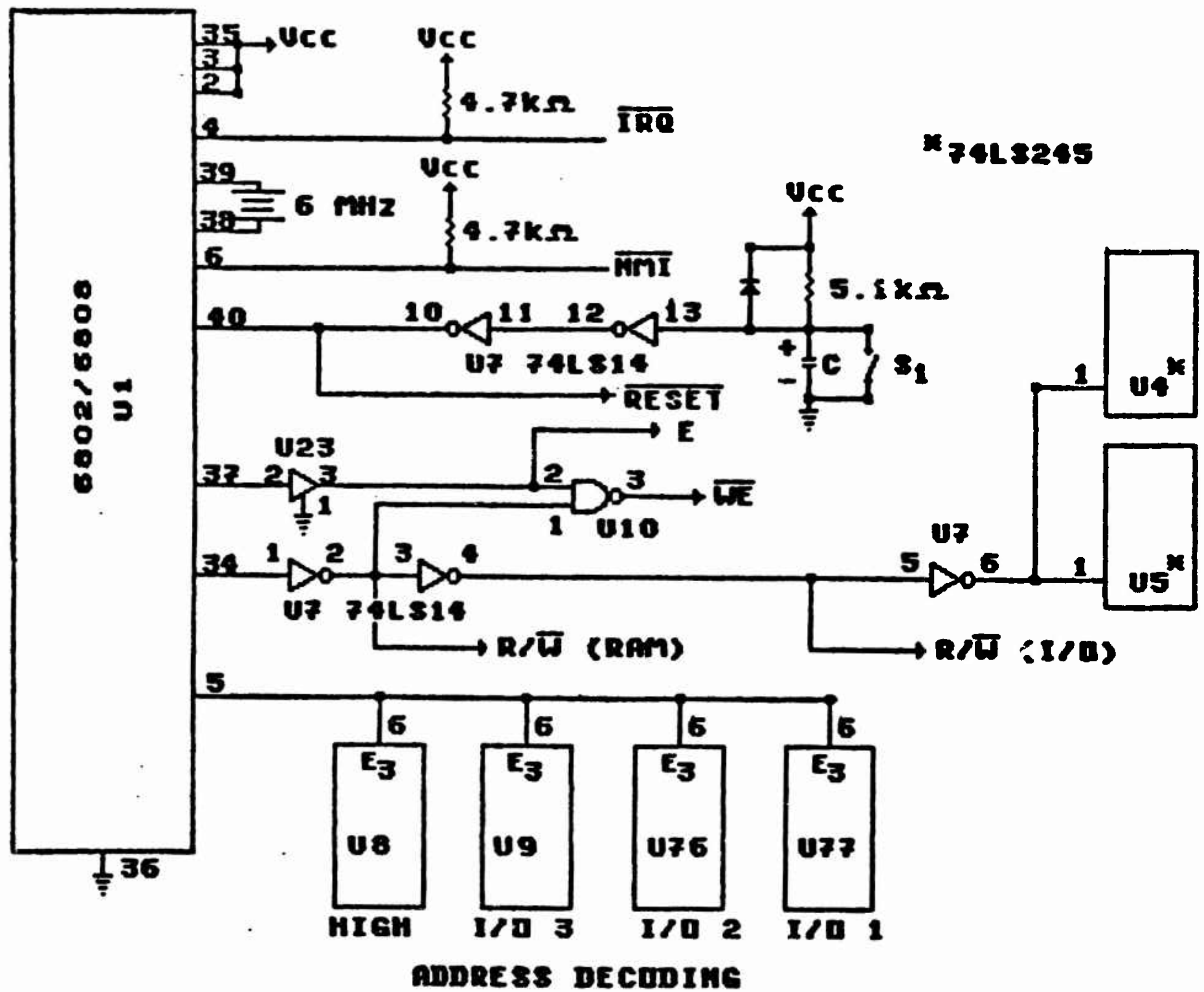
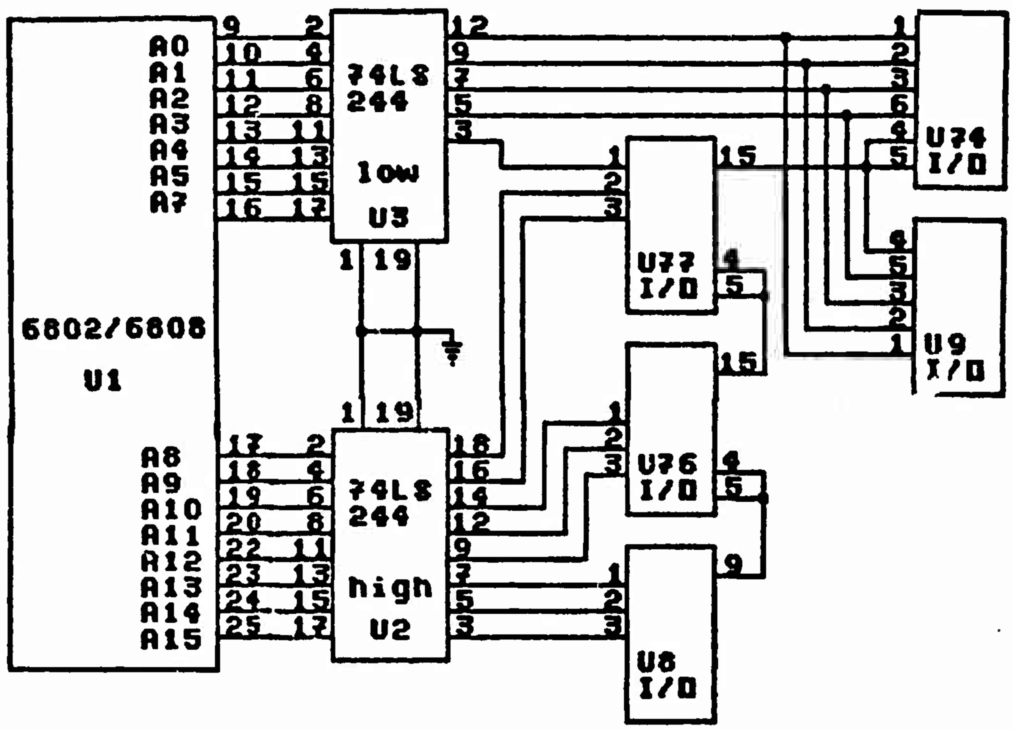


Fig. G.1 CPU Control Bus



U2	74LS244	HIGH ADDRESS
U3	74LS244	LOW ADDRESS
U8	74LS138	8K of 64K
U9	74LS138	8 of 64
U74	74LS138	8 of 64
U76	74LS138	1K of 8K
U77	74LS138	128 of 1K

Fig. G.2 CPU Address Bus

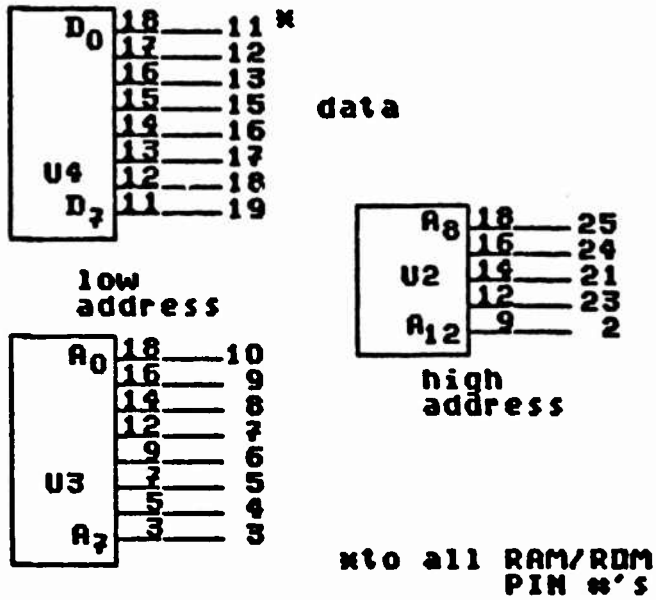


Fig. G.3 CPU Data Bus

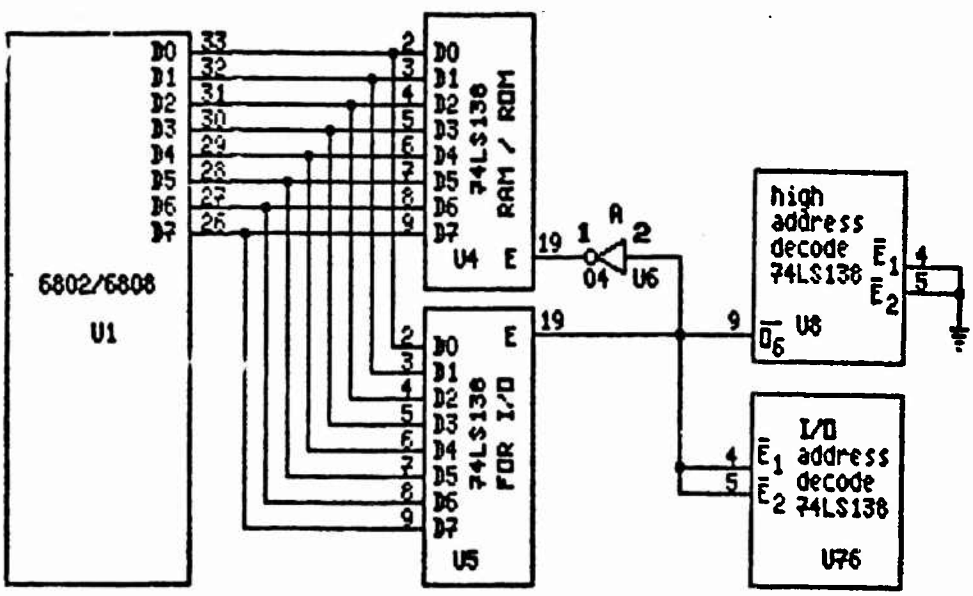


Fig. G.4 RAM/ROM Address and Data Bus

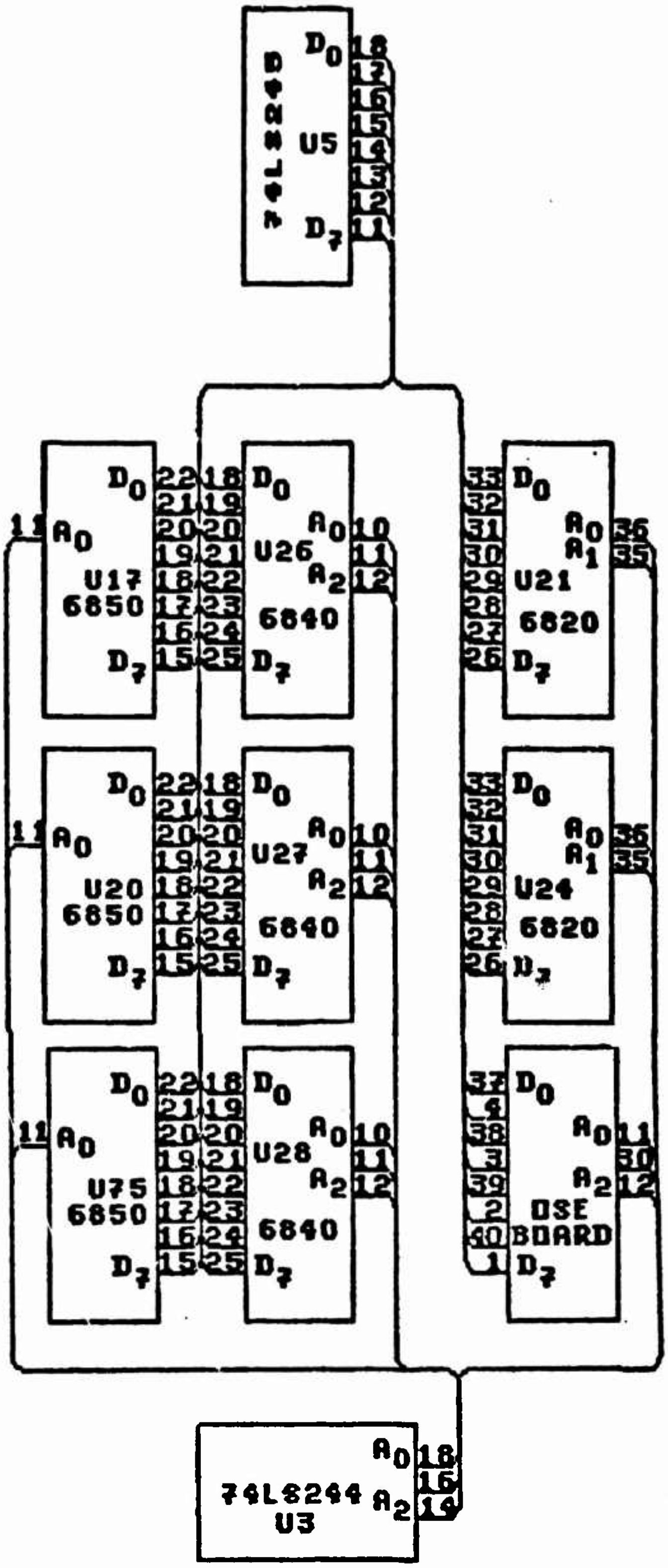


Fig. G.5 I/O Address and Data Bus

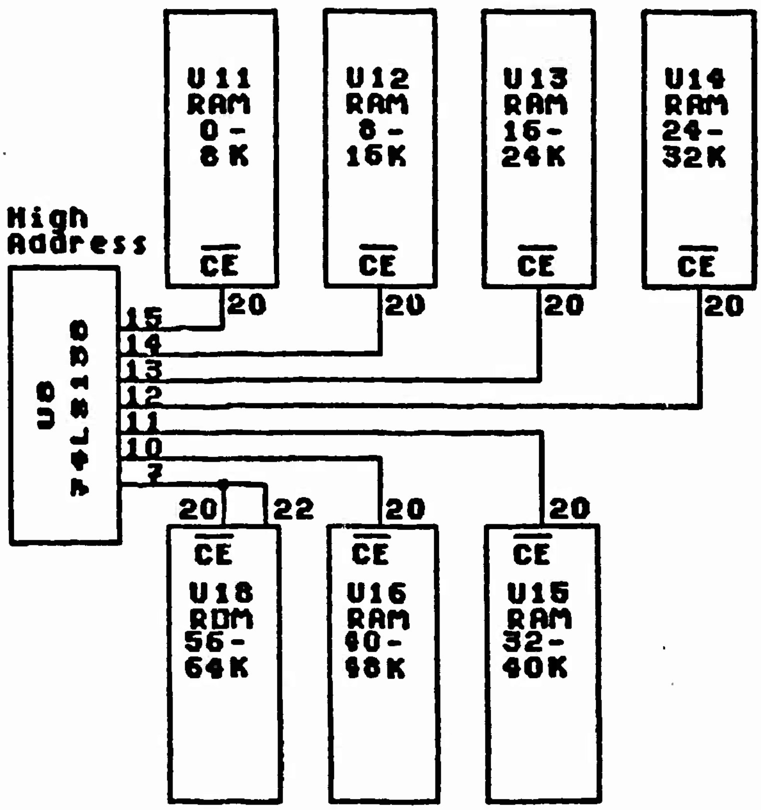


Fig. G.6 RAM/ROM Chip Enables

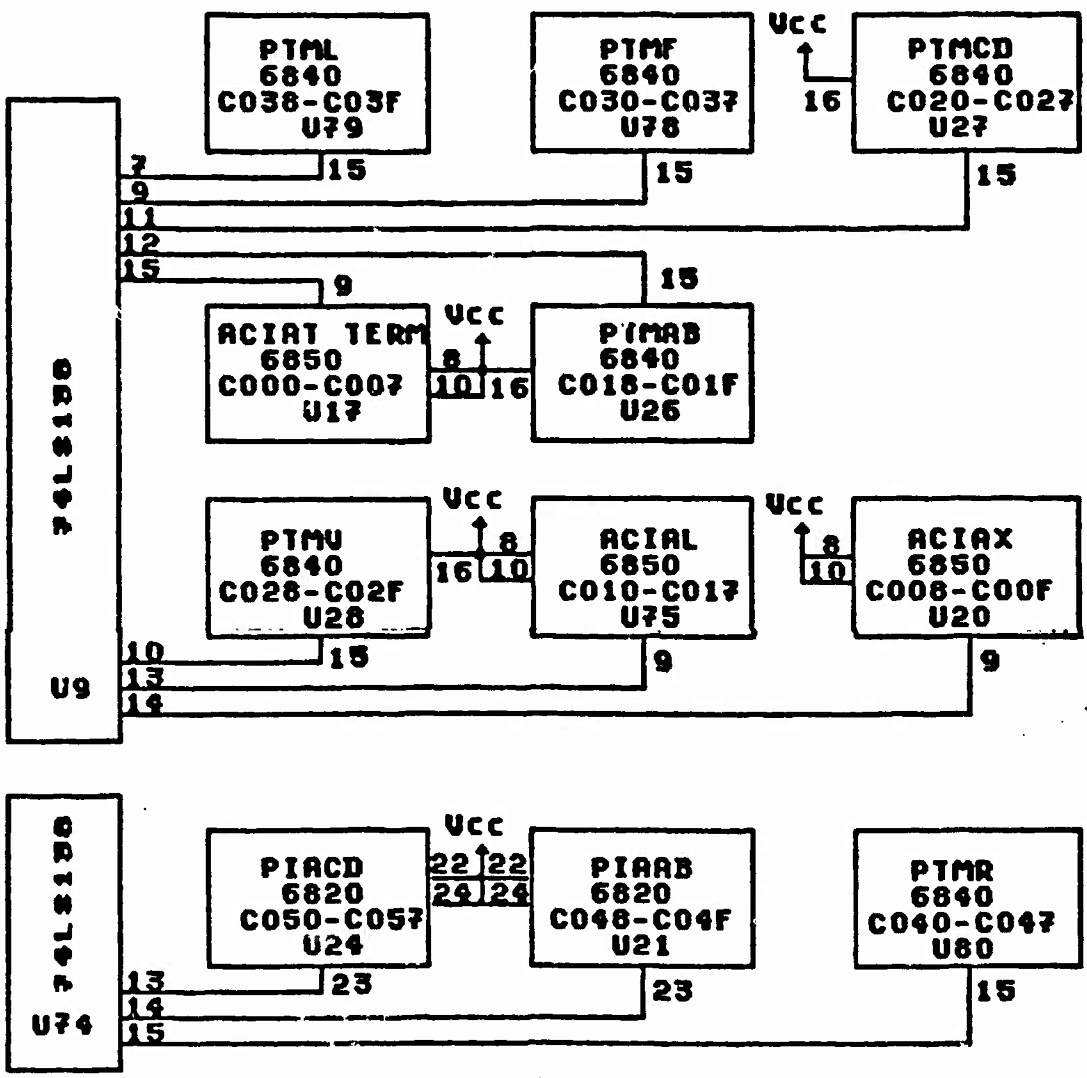


Fig.G.7 I/O Chip Enables

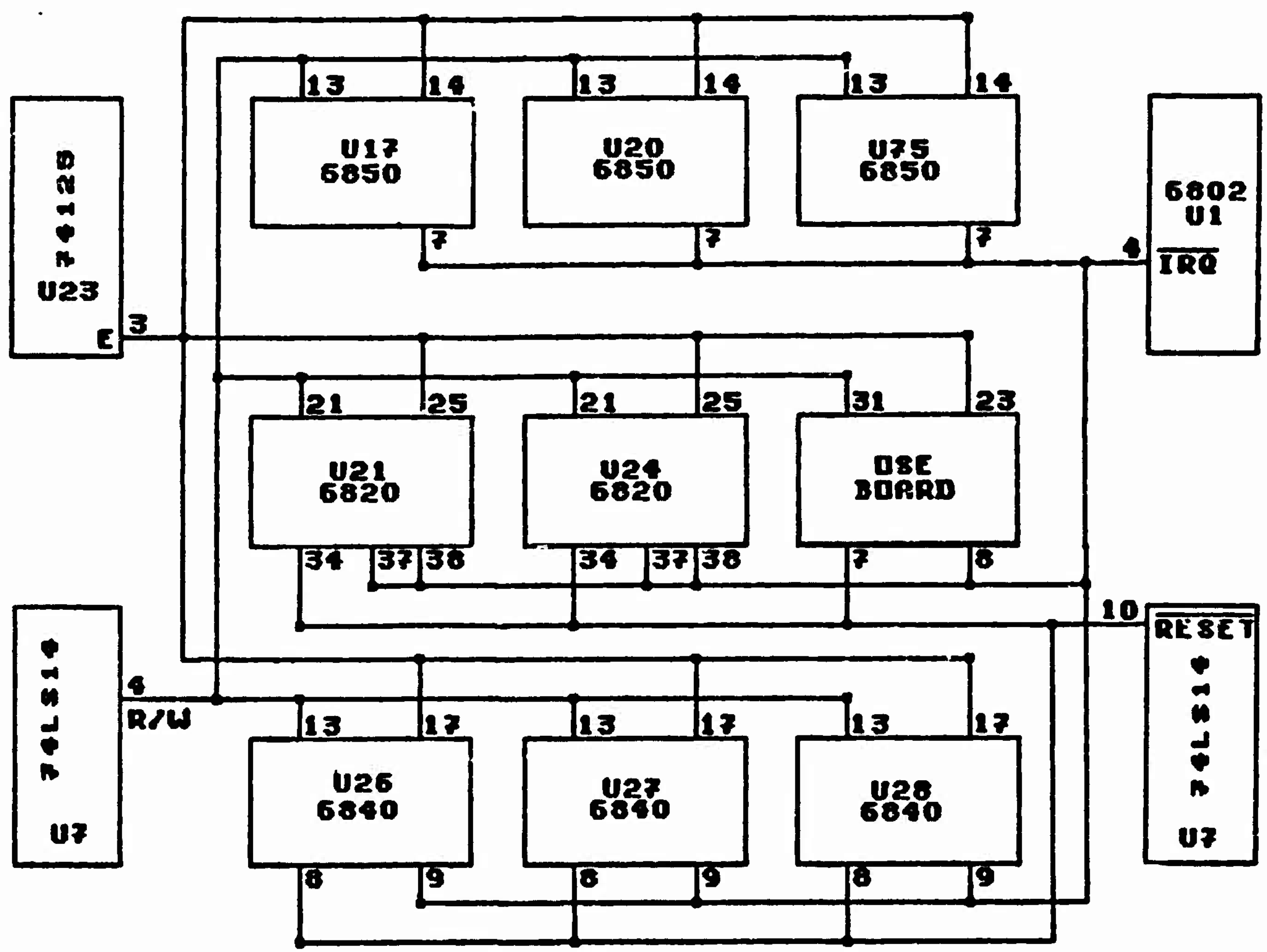


Fig. G.8 I/O Auxiliary Signals

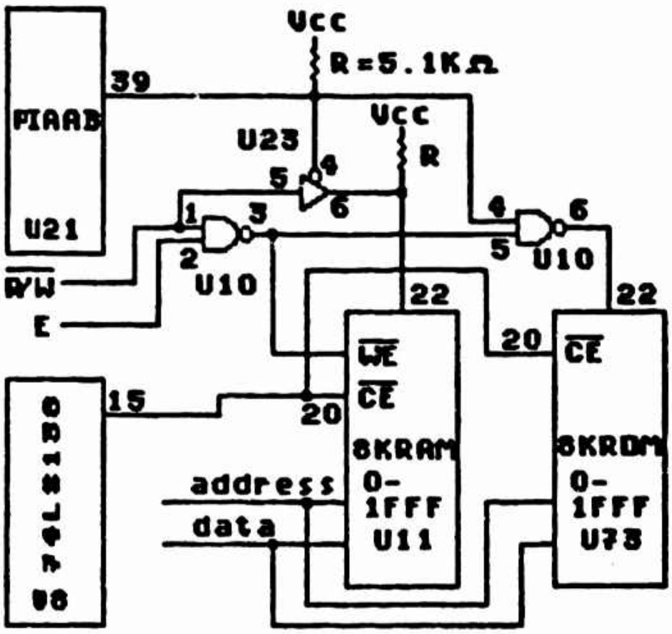


Fig. G.9 ROM Overlay RAM Circuit

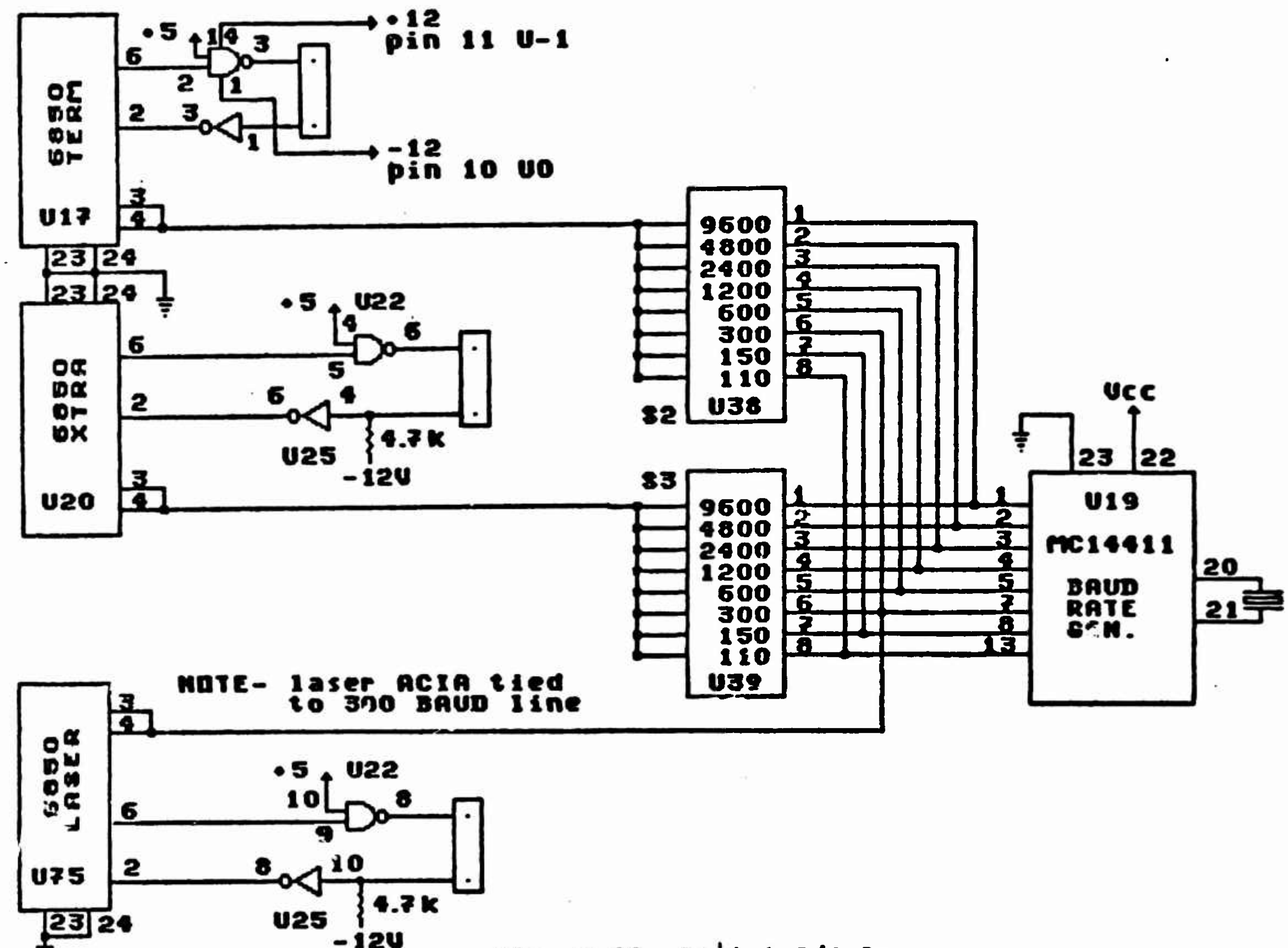


Fig. G.10 Serial I/O Ports

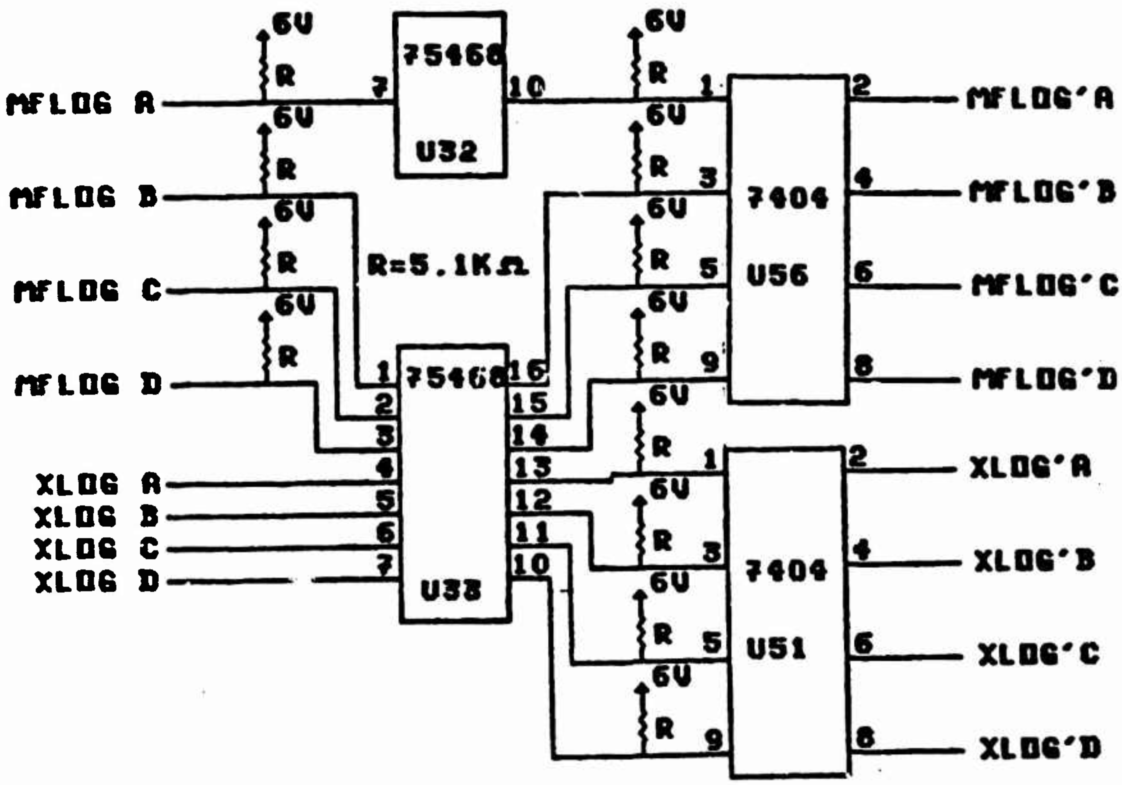


Fig. G.11 Buffered Sonar Signals

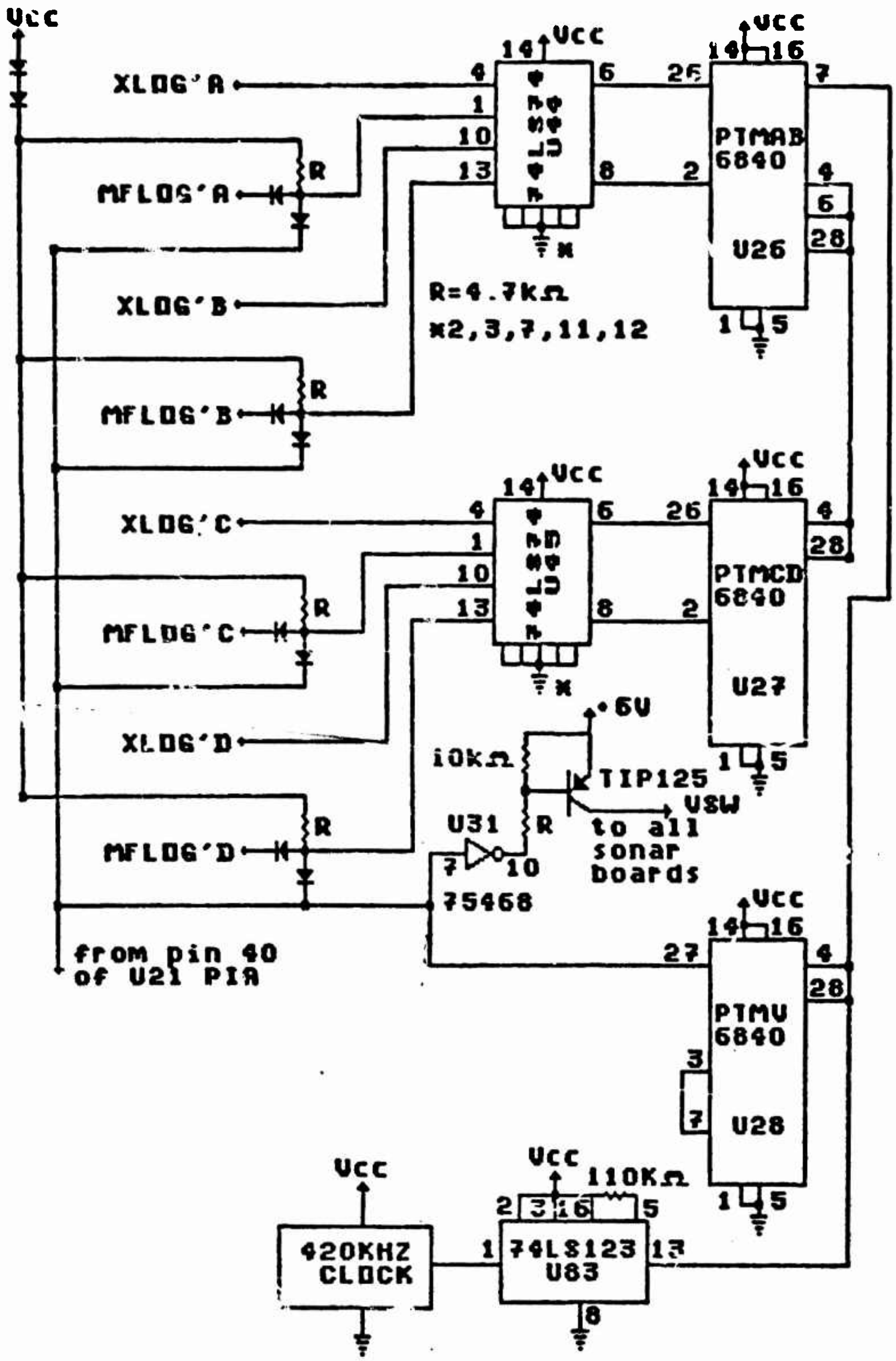


Fig. G.12 Sonar Timer/Range Board Interface

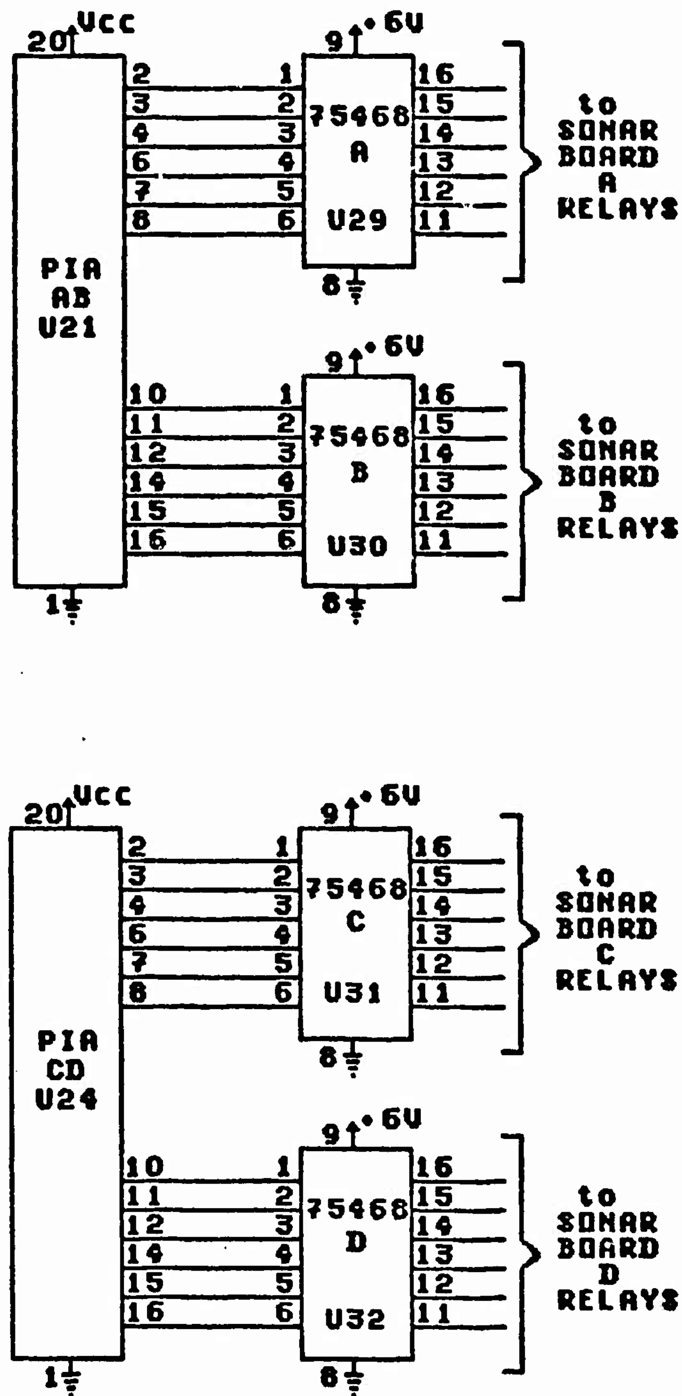
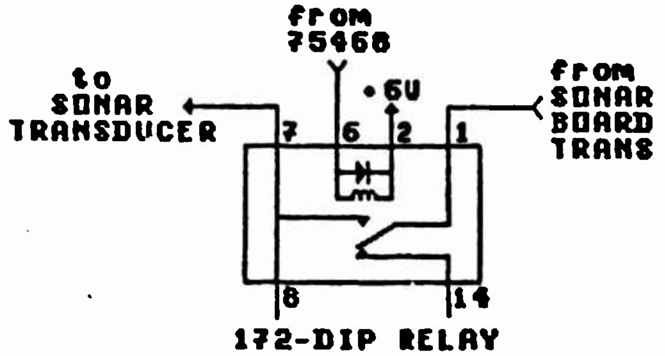


Fig. G-13 Sonar Transducer Selection Circuitry



The transmit line of each SONAR BOARD is tied to eight relays. Each relay is tied to one SONAR transducer. A shielded coaxial line is used between each relay and each transducer.

Fig. G.14 Sonar Transducer Relay Interface

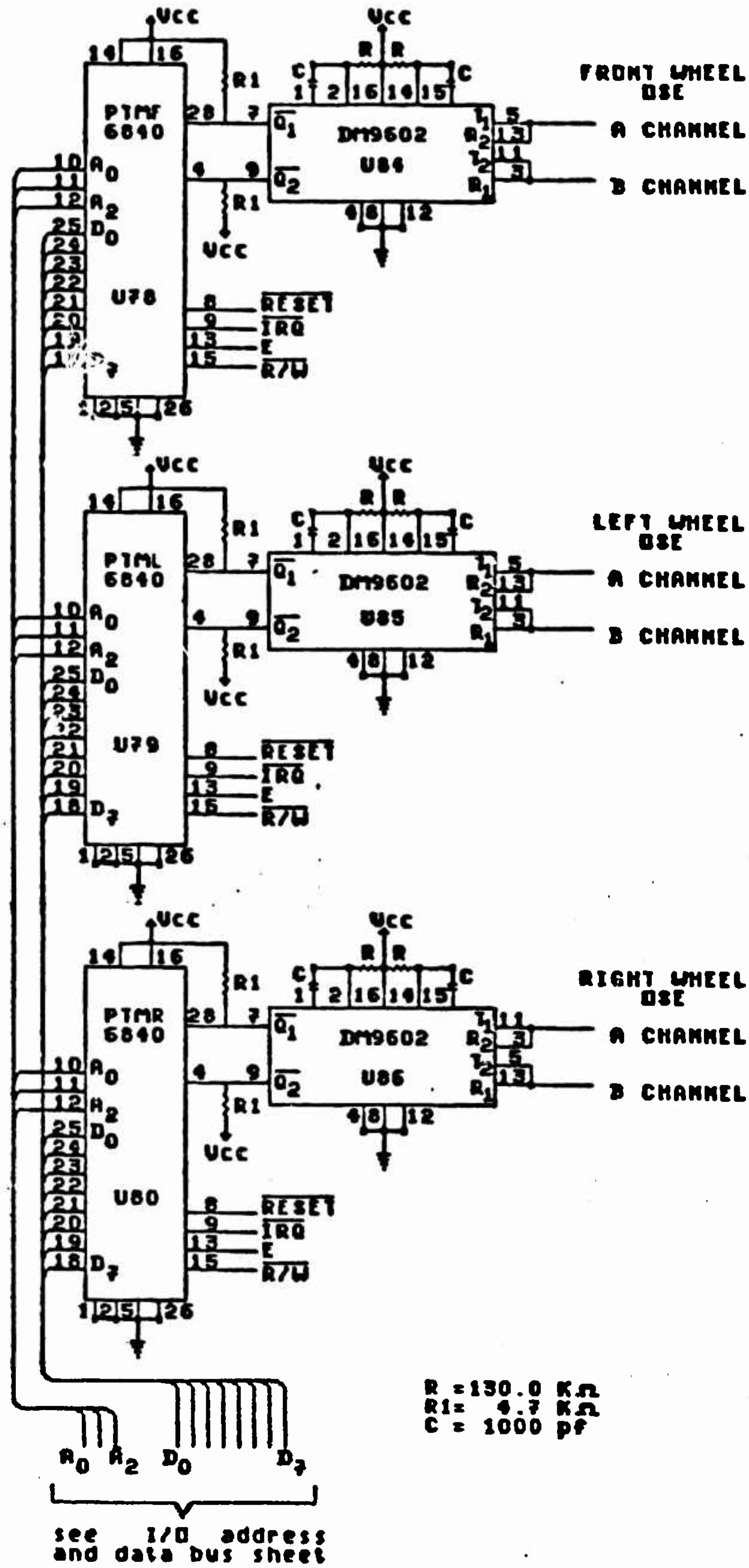


Fig. G.15 Optical Shaft Encoder Timer/Counter Circuit

C0	C1	C2	C4	C5	C6	D0	D1	D2	D4	D5	D6
13	14	15	16	17	18	19	20	21	22	23	24

12	11	10	9	8	7	6	5	4	3	2	1
B6	B5	B4	B2	B1	B0	A6	A5	A4	A2	A1	A0

Fig. G.16 24 Pin Sonar Transducer Header

VSW	6VDC	XLOG		GND	XMIT		MFLOG	6VDC	GND
20	19	18	17	16	5	4	3	2	1

1	2	3	4	5	16	17	18	19	20
GND	6VDC	MFLOG		XMIT	GND		XLOG	6VDC	VSW

Fig. G.17 Dual Sonar Range Board 20 Pin Header

D7	1	40	D6
D5	2	39	D4
D3	3	38	D2
D1	4	37	D0
	5	36	
	6	35	
RESET*	7	34	
IRQ*	8	33	
	9	32	
	10	31	R/W
A0	11	30	A1
A2	12	29	
	13	28	
	14	27	
	15	26	
	16	25	
	17	24	
	18	23	E
GND	19	22	
5VDC	20	21	

Fig. G.18 Optical Shaft Encoder Board 40 Pin Header

APPENDIX H

HERO-I TO MARRS-1 CONVERSION

CONTENTS OF APPENDIX H

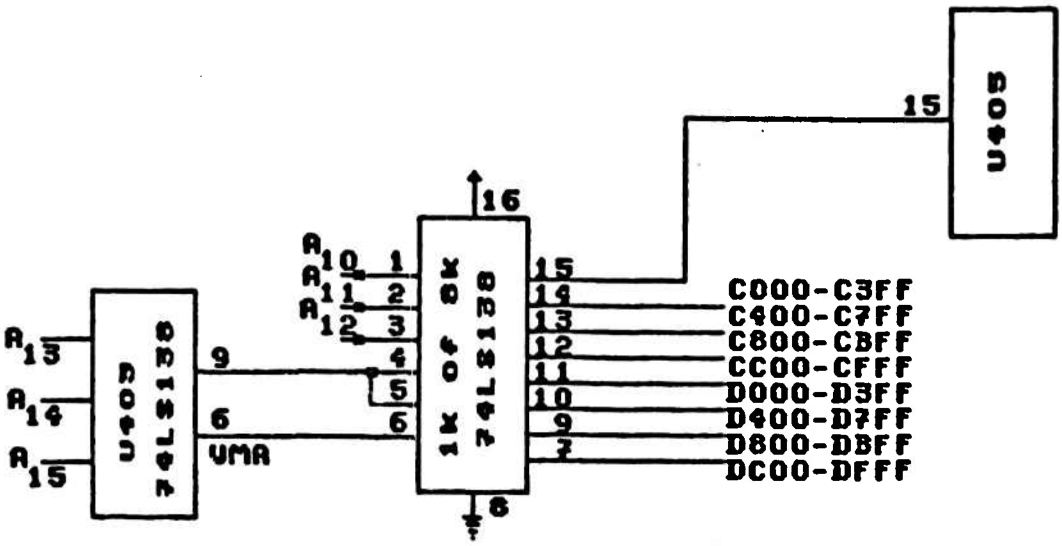


FIG. H.1 DRIVE COMPUTER I/O UPGRADE

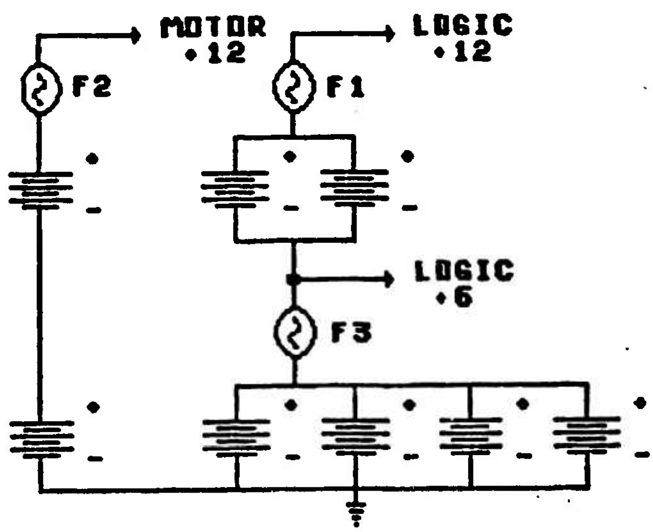


FIG H.2 MARRS-1 BATTERY CONFIGURATION

APPENDIX J

AFIT MOBILE ROBOTICS LABORATORY EQUIPMENT,

COMPUTER HARDWARE, AND SOFTWARE

EQUIPMENT

QUANITY	DESCRIPTION
1	AFIT MARRS-1 ROBOT
2	WORK BENCH
1	SMALL TABLE
2	CHAIR WITH WHEELS
3	CHAIR
2	CHAIR, HIGH
1	TABLE, FOLDING
1	TABLE, EQUIPMENT
1	LAMP, DESK
2	LAMP, WORK BENCH
1	FAN
2	TRASH CAN
2	PAPER HOLDER, TYPIST
1	HP6236B POWER SUPPLY
1	BECKMAN 3010 UL DIGITAL VOM
1	BK 1570A OSCILLOSCOPE
1	ANDERSON-JACOBSON MODEM AD342
1	HP 1610A LOGIC STATE ANALYZER
1	RACAL-VADIC VA3414 1200/300 BAUD MODEM
2	STANDING BOOK CASE

COMPUTER HARDWARE

MARRS-1 NAVIGATION COMPUTER

MARRS-1 DRIVE COMPUTER

(MENOS I UPGRADE TO HERO COMPUTER BY VIRTUAL DEVICES)

HEATH H-89 COMPUTER

H-27 8 INCH DISK DRIVE SYSTEM

TRS 80 COLOR COMPUTER (64K EXTENDED BASIC)

5.25 INCH DISK DRIVE

TRS 80 COLOR MONITOR

TRS 80 CGP 115 COLOR GRAPHIC PRINTER

TRS 80 MULTIPACK INTERFACE

TRS 80 X PAD MODEL GT116

MOTOROLA 6800 EXORCISER COMPUTER SYSTEM

HEATH H125 DOT MATRIX PRINTER

COMPUTER SOFTWARE

WORDSTAR (EDITOR) BY MICROPRO INTERNATIONAL

MENOS I VAMP (ROBOT PROGRAMMING LANGUAGE BY VIRTUAL DEVICES)

ROBO C (C CROSS COMPILER FOR HERO ROBOT BY VIRTUAL DEVICES)

ROBO A (CROSS ASSEMBLER FOR HERO ROBOT BY VIRTUAL DEVICES)

MODEM 720 (PUBLIC DOMAIN COMMUNICATION PROGRAM)

TRS 80 EXTENDED COLOR DISK BASIC

OS-9 (OPERATING SYSTEM FOR TRS 80 COLOR COMPUTER)

BASIC09 (BASIC RUNNING UNDER OS-9)

APPENDIX K

datametrics

Datametrics Inc., 340 Fordham Road, Wilmington, Mass 01887 • Tel (617) 658-5410 TWX 710-347-7672

K3 SERIES KIT ENCODER INSTALLATION INSTRUCTIONS (SQUAREWAVE VERSION)

1. MOUNTING ARRANGEMENT

Refer to Datametrics Dwg. B02537 for mounting surface arrangement. Note that (2) #4-40 X .18 deep tapped holes are called for. The two tapped holes are used to mount the kit encoder to the motor surface with #4-40 X 5/16 socket head screws.

Two #2-56 X 3/4 pan head screws are used to hold down the encoder cover. These screws thread into the encoder housing at the places indicated.

Note that an alternative mounting arrangement is possible. The two #4-40 mounting screws may be replaced by three #2-56 X 5/16 socket head screws located at the positions indicated on Dwg. B02537. In this case note that the two holes that are diametrically opposite in the encoder housing will have oversize clearance for #2-56 hardware. This clearance is reduced to appropriate size using the two fibre shoulder washers supplied with the unit.

2. INSTALLATION PROCEDURE

Step 1- INSTALL ROTOR/HUB ASSEMBLY
Slide rotor/hub assembly onto the shaft. A close sliding fit is desired. Excessive looseness can cause high rotor wobble. Slide the assembly to a position which will allow the rotor to enter the photohead gap with safe clearance both above and below the rotor disc. The bottom of the disc surface will be approximately 1/4" above the encoder mounting surface. Lightly tighten one hub set screw.

Note that the #4-48 hub set screws are much easier to handle if the hex wrench is held in a pin vise.

Step 2- INSTALL PHOTOHEAD ASSEMBLY
Slide the photohead assembly along the motor mounting surface so that the disc enters the photohead gap. Find the mounting hole locations and drop in the mounting screws. Be sure there is safe clearance above and below the disc. Tighten down the mounting screws. Note that the mounting hardware is easier to handle if a ball-tip Allen wrench is used.

Loosen the hub set screw and set the working air gap using the blue plastic shim. Insert the shim along the right side of the horseshoe opening as the opening faces you. Place the shim on top of the stator located on the upper right side of the horseshoe opening.

Step 3- ELECTRICAL CONNECTIONS
See Table 1.

Step 4- TEST ENCODER
Run the shaft at the desired speed and functionally test the encoder. The output signals' dc balance (symmetry) and quadrature phase relationship are factory set and should ordinarily not require adjustment. However, should finer trim be desired continue as follows: Adjustment pots are accessible on the photohead printed circuit board which allow a fine trim of the symmetry of the output signals. See Figure 2. The phase relationship of the output signals can be changed by movement of the photohead assembly on the mounting surface within the range of clearance around the mounting hardware. Both symmetry and phase adjustments described above can only be achieved while observing the output waveforms with an oscilliscope. See Figure 1.

CAUTION

ALWAYS APPLY DOWNWARD PRESSURE ON THE PHOTOHEAD ASSEMBLY WHILE ATTEMPTING PHASE ADJUSTMENT IN ORDER TO AVOID RUBBING THE DISC AGAINST THE STATOR.

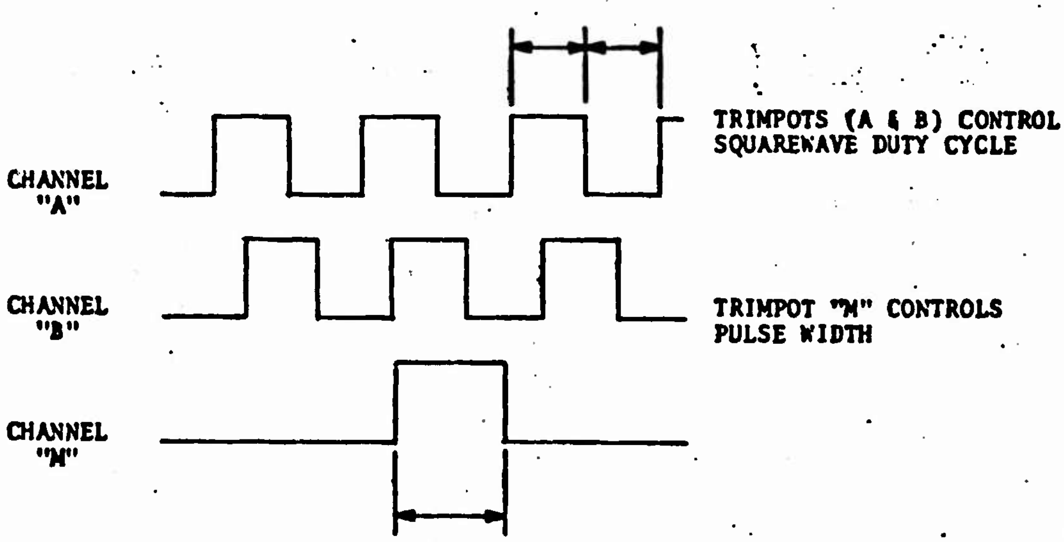


Figure 1 - Waveform Characteristics
(CCW rotation viewed from encoder end)

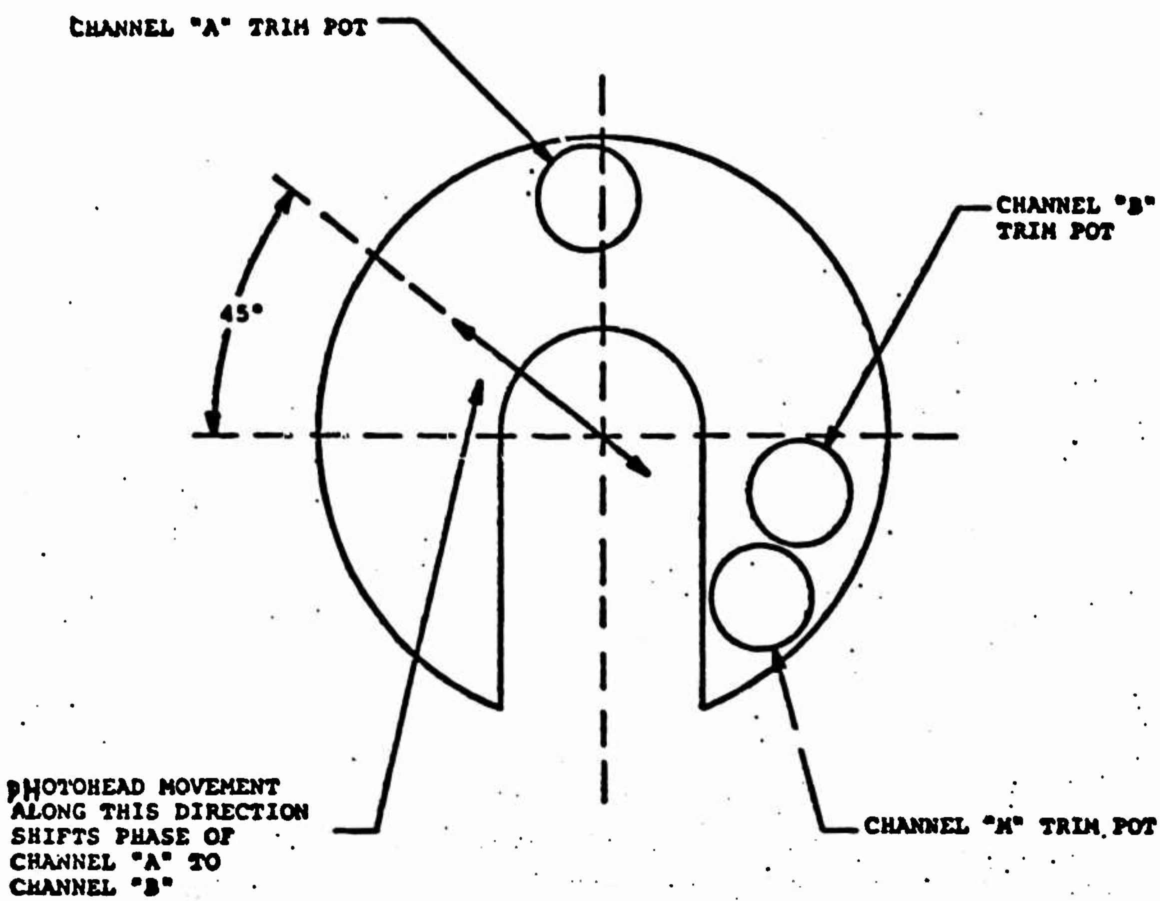


Figure 2 - Photohead Board
Pot Layout

NOTE: The trim pots are factory set and glyptolled. They ordinarily should not require re-adjustment.

Step 5 - INSTALL COVER
Install cover over encoder using (2) #2-56 screws provided. Do not over tighten cover screws.

Table 1 - Electrical Connections

Wire Color	Function
Yellow	Output A
Blue	Output B
Orange	Output M
Red	+V
Black	Ground

POLAROID ULTRASONIC RANGING SYSTEM

Unmodified Ranging Circuit Board #606191
Modified Ranging Circuit Board #606745

TECHNICAL DATA

General Description:

The ranging modules contains all necessary components to: generate the drive signal for the transducer; control timing functions; receive, amplify and filter the returned echo; and process this signal providing a step function output at the time of the received echo. The distance from the transducer to the target can then be computed with additional circuitry, knowing the speed of sound in air (or other gas), and the time interval between the transmit signal and the received echo as provided by the ranging module.

The #606745 board supplied with the designer's kit consists of a #606191 board which has had a six-wire, ribbon cable attached as shown on Figure 2 and has been modified as shown on Figures 4 and 5. The #606745 board does not have the transducer cable attached; however, this cable can be purchased from Polaroid by requesting Cable Assembly #604789.

Features:

Measurement range of 0.9 feet to 35 feet

Nominal resolution ± 0.12 inches to 10 feet
± 1% over entire range

Multiple measurement capability

Drives Polaroid Electrostatic Transducer which requires 50kHz 300V signal with no additional interface.

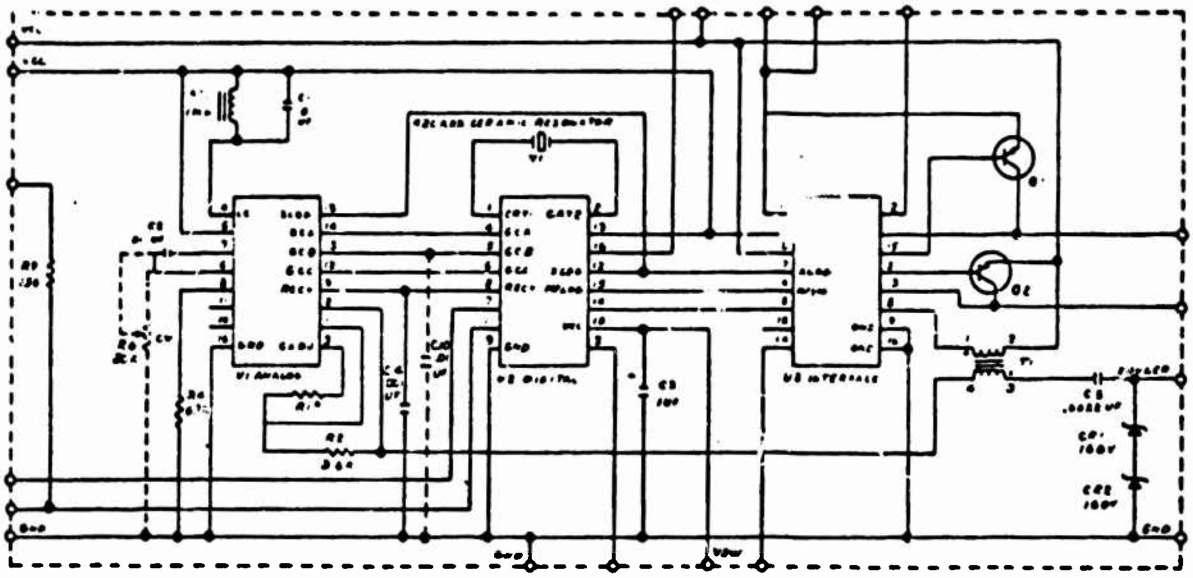
Designed to operate with the Polaroid Instrument Grade Electrostatic Transducer #604142.

Operation Conditions:

Supply Voltage	5.0 Vdc
Continuous Operating Current	250 mAmp
Peak Current During Transmit	2.5 Amp
Temperature Range	0 to 40 Deg. C.

Unmodified Ranging Circuit Board #606191
Modified Ranging Circuit Board #606745

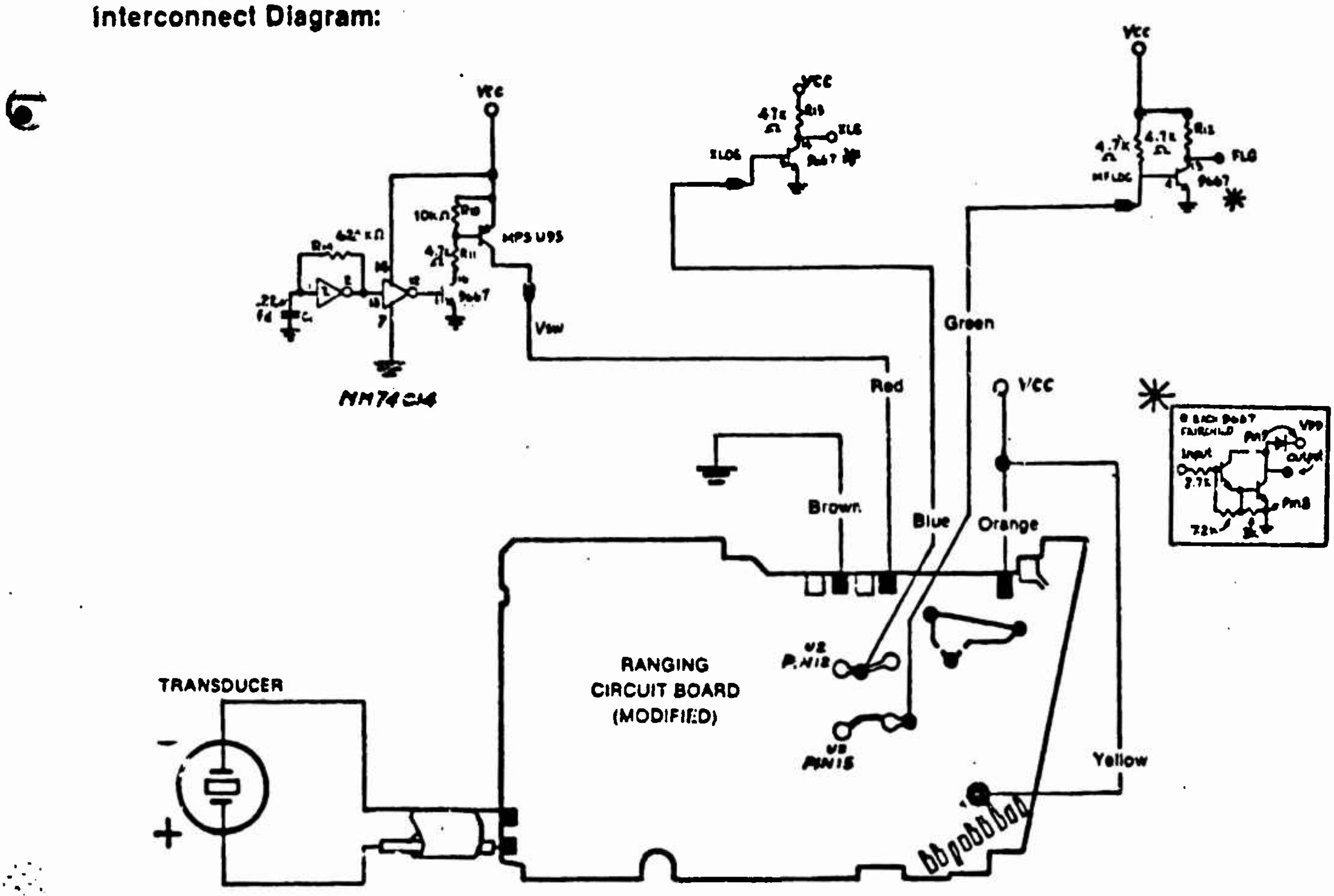
Schematic:



NOTES: 1. R1 is selected to set gain 2. R6 & C10 are only added if necessary at factory

FIG. 1

Interconnect Diagram:



NOTE: Color Codes are for the ribbon cable supplied with #606745 Modified Ranging Circuit Board.

FIG. 2

Unmodified Ranging Circuit Board #606191
Modified Ranging Circuit Board #606745

Signal Description:

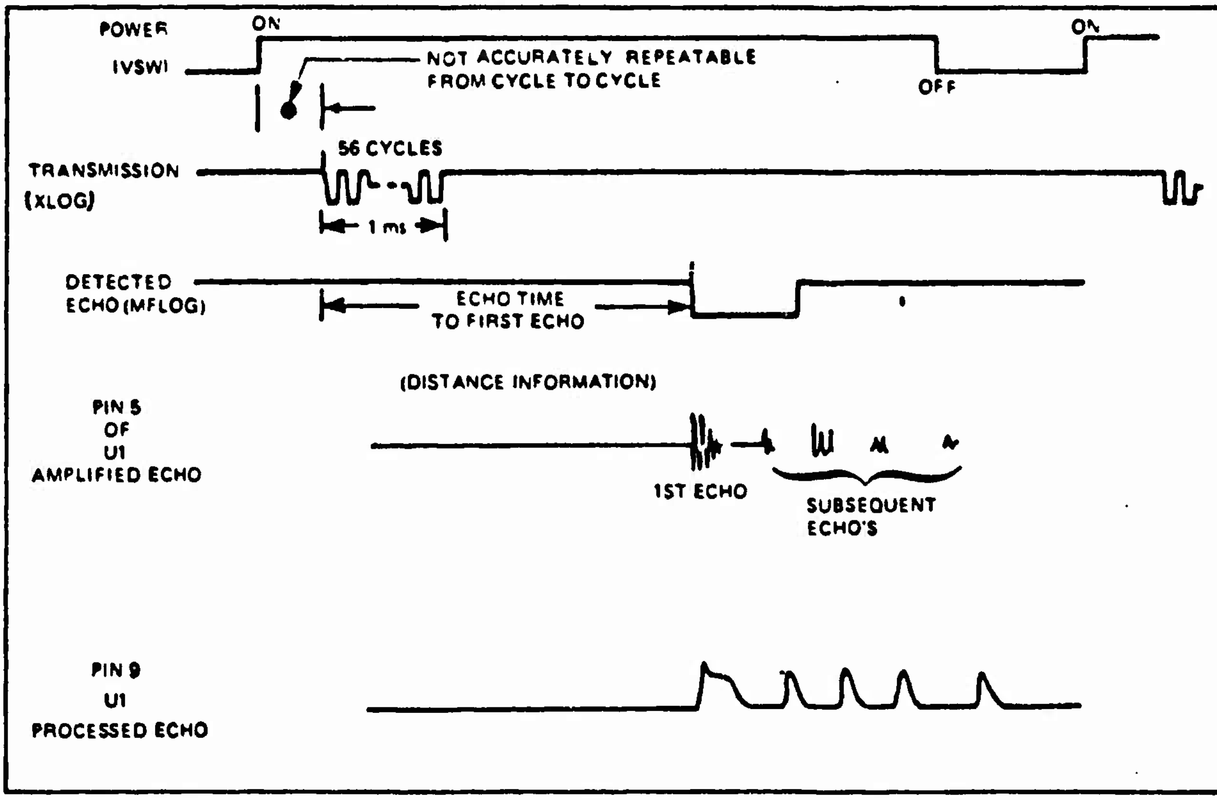
NAME	DESCRIPTION	Wire Color On #606745
VCC	Positive, Power Supply	Yellow & Orange
VSW	Starts a transmit/receive cycle when taken high Resets for next cycle when taken low	Red
XLOG	Transmit detect, use first falling edge after VSW high	Blue
MFLOG	Receiver detect, goes low when an echo is detected	Green
GND	Ground, power supply	Brown
XDUCER	Transducer input	—
GND	Transducer ground	—

Electrical Characteristics:

PARAMETER	MIN.	TYP.	MAX.	UNITS
Supply Voltage	4.9		6.8	Vdc
Continuous Operating Current			250	mAmp
Peak Current (during transmit)		2.5		Amp
VSW				
Vih	4.0		Vcc	Vdc
Iih	100			mA
Vil			0.4	Vdc
XLOG				
Voh	2.0			Vdc
Vol			0.8	Vdc
Iol			0.5	mA
MFLOG				
Voh	2.0			Vdc
Vol			0.8	Vdc
Iol			1.0	mA

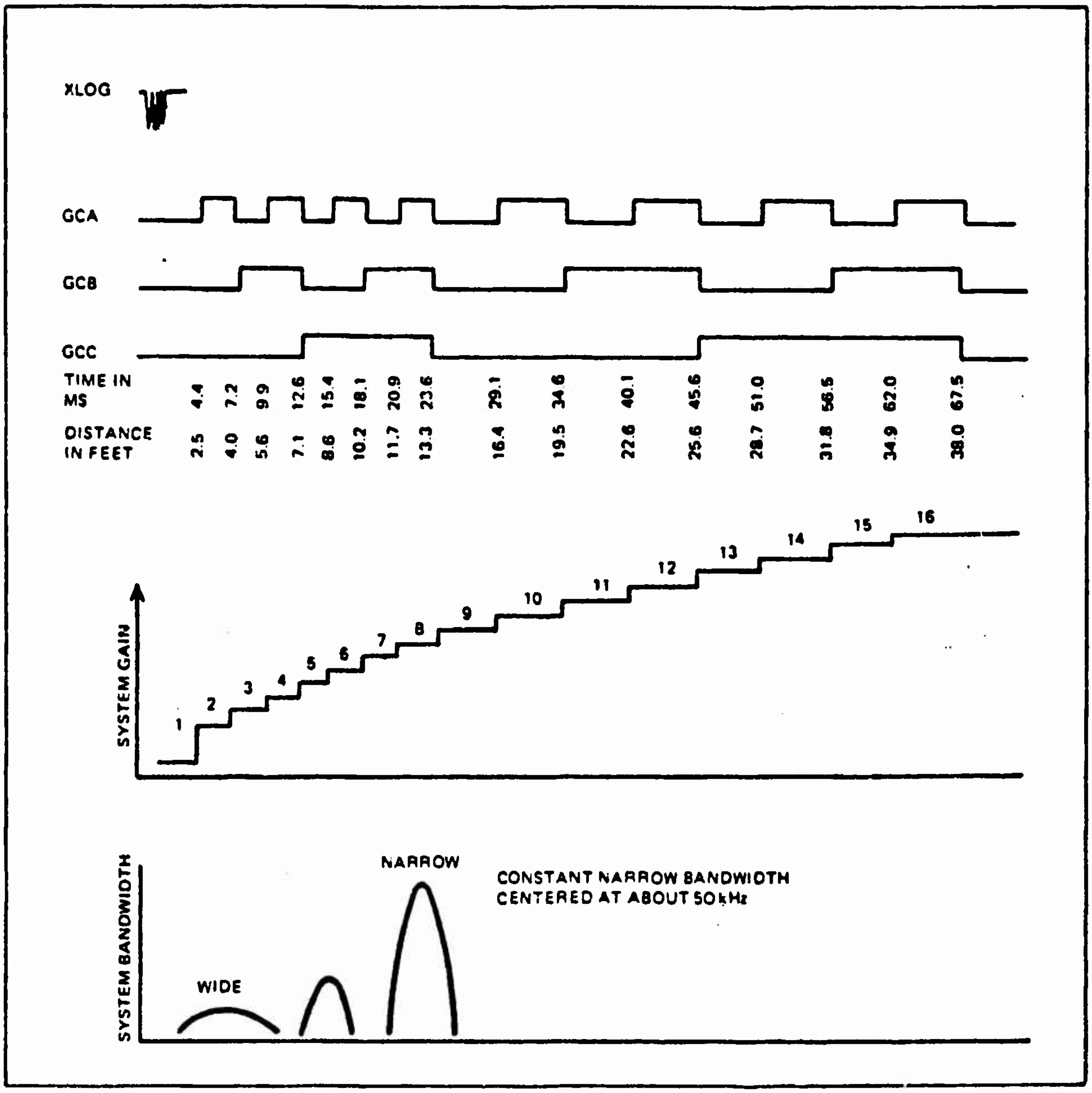
Unmodified Ranging Circuit Board #606191
Modified Ranging Circuit Board #606745

Timing:



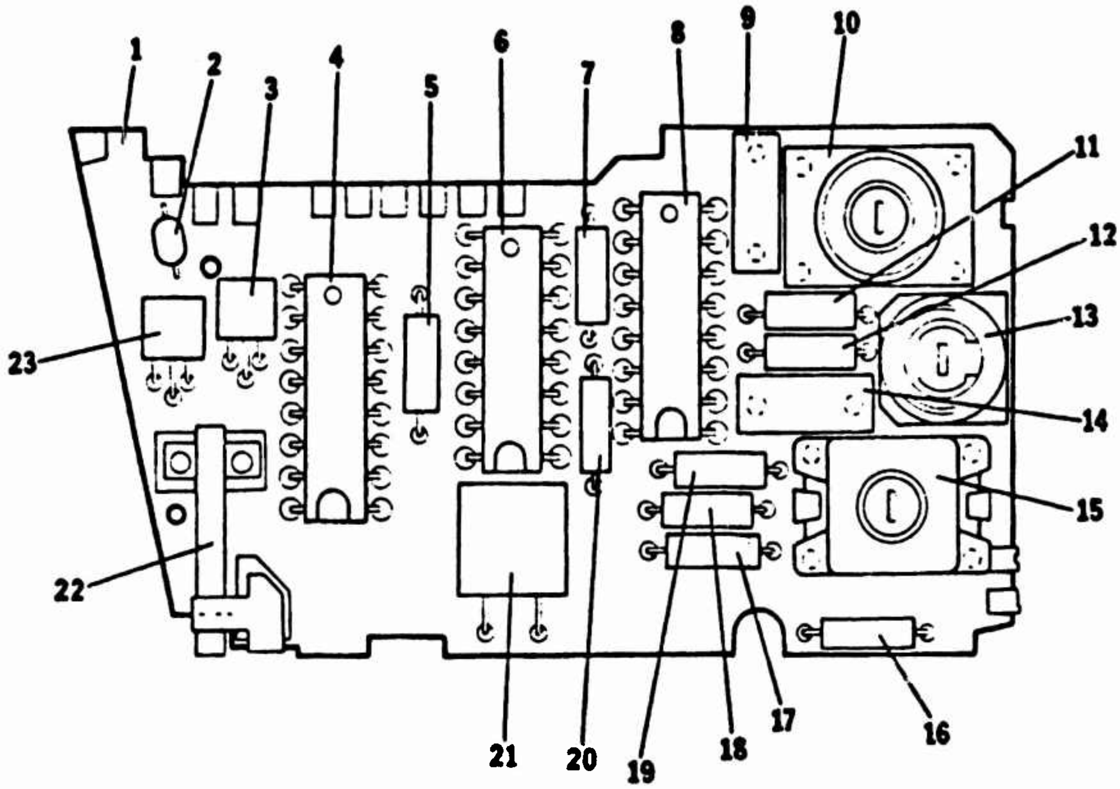
Unmodified Ranging Circuit Board #606191
Modified Ranging Circuit Board #606745

Typical Relative Gain Bandwidth:



Unmodified Ranging Circuit Board #606191
Modified Ranging Circuit Board #606745

Parts List:



ITEM	REF.	DESCRIPTION
1		P.C. BOARD
2	C3	CAPACITOR 1μf, 35V
3	Q2	TRANSISTOR
4	U3	POWER IC
5	R9	RES. 150 Ω
6	U2	DIGITAL IC
7	C4	CAPACITOR .001μf, 10V
8	U1	ANALOG IC
9	C1	CAPACITOR .01μf, 10V
10	L1	TUNED CIRCUIT INDUCTOR
11	C2	CAPACITOR .01μf, 10V
12	R4	RESISTOR (62K, 5%, ¼W)
13	R6	VARIABLE RESISTOR (25K, ¼W) (See Note 2)
14	C5	CAPACITOR .0022μf, 400V
15	T1	TRANSFORMER
16	R1	RESISTOR (1-10K, 5%, ¼W) See Note (1)
17	CR1	DIODE ZENER (IN4006)
18	CR2	DIODE ZENER (IN4006)
19	R2	RESISTOR (130K, 5%, ¼W)
20	C10	CAPACITOR .01μf, 10V See Note (2)
21	XTAL	CRYSTAL 420 kHz
22	S1	SWITCH
23	Q1	TRANSISTOR

NOTE: 1. Resistor Value Set at Factory

NOTE: 2. Only Added If Necessary at Factory

Unmodified Ranging Circuit Board #606191
Modified Ranging Circuit Board #606745

Dimensions

SIZE MAX DIM.
3.025 IN. X 1.652 IN.
X 0.520 IN.

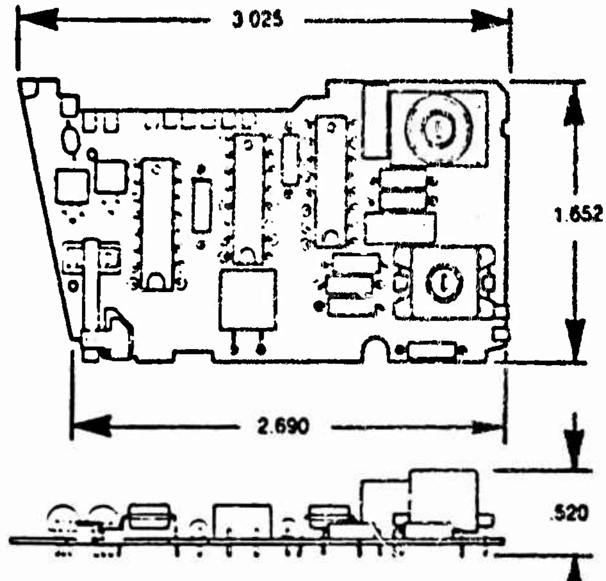


FIG. 3

Application Notes:

The ranging modules #606191 and #606745 are designed to function with the Polaroid Instrument Grade Transducer #604142.

Power the module through VCC. Next bring VSW high. The transmit burst of 56 pulses will begin approximately 5 msec later. The 56 pulses consist of 8 cycles at 60 kHz, 8 cycles at 56 kHz, 16 cycles at 52.5 kHz and 24 cycles at 49.41 kHz and lasts for a period of about 1.0 msec. Detected transmit can be observed at U2-12. The receiver is blanked for 1.6 msec. If an echo is not detected within 62.5 msec MFLOG will go high. Also if an echo is detected before 62.5 msec and after 1.6 msec MFLOG will go high.

Range information is determined by the time interval between the first falling edge of XLOG and the falling edge of MFLOG. The speed of sound in air is:

$$C = 331.4 \sqrt{\frac{T}{273}} \quad \text{M/Sec}$$

C = Speed of sound in air

T = Temperature in degree Kelvin

(Kelvin = Celsius + 273)

M = Meters

Sec = Seconds

To initiate another range measurement bring VSW low then high again while leaving VCC on.

The system (transducer and ranging module) gain is set to detect a 1.34 inch diameter sphere at 4 feet 3 inches on the acoustic axis.

If the ranging circuit board contains R6, adjustments are not recommended. R6 can change system gain and will change these specifications. If R6 is needed, refer to the parts list for installation location and specification. C2 will have to be relocated to the unused pad connected to the R6 wiper.

Unmodified Ranging Circuit Board #606191
Modified Ranging Circuit Board #606745

Modifications:

The Ultrasonic Circuit Board (#606745) included with the Ultrasonic Designer's Kit has been modified for use in ranging applications. Module #606191, ordered separately, will have to be modified by the purchaser to achieve the same performance. Other changes could be necessary for a particular application. Refer to the figure below to see how the ranging module #606191 must be altered to obtain performance similar to that of module #606745 in the kit. Make the alterations as follows:

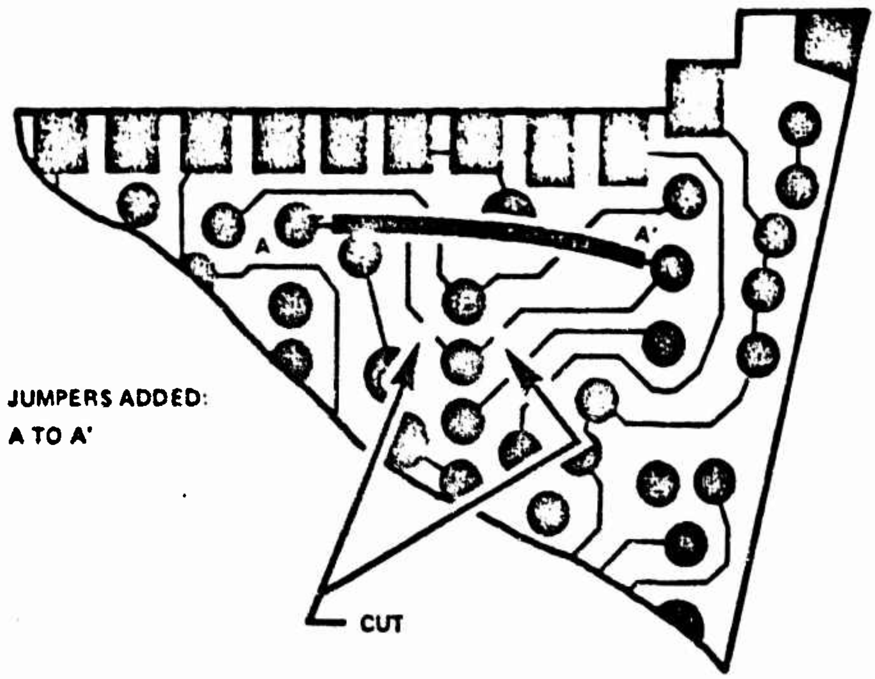


FIG. 4

A. Cut the metallic circuit path on the board at the two points indicated by the arrows. Scrape the metal away or use any other effective method.

B. Solder 1 jumper wire to the board as shown: between points A and A'.

C. Cut switch as shown.

FIG. 5

For additional information or technical assistance contact:

Polaroid Corporation
Battery Division
784 Memorial Drive - 4A
Cambridge, Massachusetts 02139
(617) 577-4681

BD-263 "Polaroid" and "Polapulse"®
Printed in U.S.A. 7/83

Appendix M

6800 Hex File Format

The 6800 Hex file format provides a compact representation of binary data patterns for transmission using ASCII communication terminals.

The Hex fileis organized into data records with each record containing information in the same format. The record information consists of type, length, address, data and checksum. All records begin with an 'S' character for start of record identification. All information in the file which is not between a start of record and the checksum is ignored.

TAPE FORMAT

ASCII Character	Description
1	Start of record (S)
2	Type of record 0 — Header record 1 — Data record, 9 — End of file record.
3—4	Byte Count Since each data byte is represented as two hex characters, the byte count must be multiplied by two to get the number of characters to the end of the record. (This includes checksum and address data.)
5, 6, 7, 8	Address Value The memory location where this record is to be stored.
9, ..., N	Data Each data byte is represented by two hex characters.
N+1, N+2	Checksum The one's complement of the additive summation (without carry) of the data bytes, the address, and the byte count.

Example Data Record

Memory Contents

Address	Data
A000	10
A001	1A
A002	20
A003	2A

Data Record Contents

Character		Tape			
1	Start of record	53	S		
2	Type of record	31	1		
3	Byte count	30	0	Checksum	
4		37	7		
5		41	A		Byte Count *2
6		30	0		
7	Address	30	0		
8		30	0		
9	Data byte 1	31	1		
10		30	0		
11	Data byte 2	31	1		
12		41	A		
13	Data byte 3	32	2		
14		30	0		
15	Data byte 4	32	2		
16		41	A		
17	Checksum	38	8		
18		34	4		

The format for all hexfile records is diagrammed below.

Character		Header Record		Data Record		End-of-File Record	
1	Start of Record	53	S	53	S	53	S
2	Type of Record	30	0	31	1	39	9
3	Byte Count	31	12	31	16	30	03
4		32		36		33	
5	Address (if any)	30	0000	31	1100	30	0000
6		30		31		30	
7		30		30		30	
8		30		30		30	
9	Data	34		39	98	46	FC
10		38		38		43	(Checksum)
.		34		30	02		
.		34		32			
.		35		.			
.		32		.			
.				41	A8 (Checksum)		
.				48			
N	Checksum	39	9E				
		45					

APPENDIX N

NAVIGATION COMPUTER SOFTWARE USER'S MANUAL

(EXTRACTED IN PART FROM AMERICAN MICROSYSTEMS 6800 PROTOTYPING MANUAL.)

THE NAVIGATION COMPUTER SOFTWARE IS COMPRISED OF THE MIKBUG AND PROTO OPERATING SYSTEMS, A ROM SUBROUTINE LIBRARY (RS)3, A SYSTEM HARDWARE INITIALIZER, AND A SYSTEM MASKABLE INTERRUPT HANDLER. MIKBUG IS SELDOM USED EXCEPT BY SPECIAL COMERCIAL SOFTWARE SUCH AS BASIC INTERPRETERS. THE PROTO OPERATING SYSTEM RESPONDS TO THE COMMANDS DESCRIBED IN THE FOLLOWING PARAGRAPHS. A COMMAND CONSISTS OF A ONE CHARACTER COMMAND IDENTIFIER FOLLOWED BY ADDITIONAL PARAMETERS IF NEEDED, SEPERATED BY BLANKS OR COMMAS. ALL COMMANDS END WITH A CARRIAGE RETURN. SINCE NO ACTION IS TAKEN BEFORE THE CARRIAGE RETURN, AND INPUT LINE MAY BE DELETED BY USE OF THE ESCAPE KEY. (PAPER TAPE AS A STORAGE MEDIA IS MENTIONED IN THE FOLLOWING PARAGRAPHS. IT SHOULD BE INTERPRETED AS THE DEVICE OR MEDIA WHICH IS USED TO STORE AND RETRIVE DATA IN THE ROBOT TO EXTERNAL COMPUTER SYSTEM.)

L, ADDL, ADDH, OFFSET

The Loadfile command loads data from a hex formatted file. (see Appendix M) into the user's memory between ADDL and ADDH, inclusive. The OFFSET is added to the memory address specified on the tape to form the actual memory starting address for the data stored. If a byte to be stored into memory has an address outside of the range ADDL, ADDH, it is not entered into memory, but a Delete character (H'FF) is transmitted to the terminal.

Example: L 0100 02FF FFFA

The address range in the L command is optional, and if omitted is assumed to be the full range of memory (0000—FFFF). The offset parameter is also optional, and if omitted is assumed to be zero (0000). Thus the L command with no parameters loads the tape into the memory locations specified on the tape with no offset. The offset value in the L command is a two's complement signed number, entered in unsigned hexadecimal. For example, an offset of -6 is entered as FFFA.

If an attempt is made to load non-existent memory, or ROM, the loading operation will terminate, typing out the address and the message "BAD ADR".

In operating the Load command, PROTO goes into a receive md and scans the file for the first ASCII "S", which indicates start of record. It is not necessary to position the file at the first record of a file since each record contains its own starting address.

PROTO will load data records until it encounters an end of file (EOF) record or a file error (Check Sum or illegal character). When PROTO reads a header record (start of record and address), it translates the header into ASCII characters and prints the result. The Check Sum is the binary sum of all characters in the block.

PROTO does not list the file contents as the file is being read.

When PROTO encounters an end of file record or a file error, it turns off the reader and prints "EOF" or "CKSM ERR" respectively.

P, ADDL, ADDH, OFFSET

The Put hex file command causes PROTO to create a hex file of the contents of memory between ADDL and ADDH, inclusive. Each record is created with a four-digit hex address of the starting byte of the record. This address is derived from the memory address of the byte being filed, plus the offset value, OFFSET. The offset is optional, and if omitted is assumed to be zero.

Records using this command (except the last record) contain 16 bytes of data plus the start code, byte count, address, and the checksum.

The P command does not cause an EOF record to be created so that several disjoint blocks of memory can be combined into one file.

Example: P F000 F07F 0F00

S, ADDR, BYTE1, BYTE2, – – –, BYTEN

The Set memory command writes the 8-bit data words specified by BYTE1 to BYTEN into consecutive memory locations starting at ADD.

If ADD has more than 4 (hexadecimal) characters or if any of the data bytes have more than 2 characters each, only the last 4 or 2 characters are used respectively.

Example: S 0000 86 05 97 28

Memory locations at 0000 thru 0003 are loaded as shown.

D, ADDL, ADDH

The Display memory command prints the contents of memory between ADDL and ADDH, inclusive, in hex format. Up to sixteen bytes per line are printed, preceded by the hexadecimal address of the first byte of the line.

G, ADDR

The Go command starts execution of the user program at the address specified by the input parameter. To insure that all registers contain the same information they held before the user program was interrupted, PROTO pushes into the stack the copy of the user registers that it keeps at locations BFEA-BFF2 (CC, B, A, X, P, S) then executes an RTI instruction. The user can change the initial values of the registers by changing the contents of these locations.

Example: G 300

Program will branch to address 0300 and start execution from that point.

R

The Registers command prints the contents of memory locations BFEA-BFF2 which contain the values that were in the user's C, B, A, X, P, and S registers (in that order) when the user's program was last interrupted.

M, ADDL, ADDH, DEST

The Move command copies memory from the range ADDL – ADDH, inclusive, to the RAM locations starting at DEST. This copy begins at the lower address, so if DEST lies within the range ADDL – ADDH, some of the original data will be lost, and other parts will be duplicated.

E

The End of Transmission command is used to cause an EOT character to be punched on the paper tape. After a field has been punched, an EOT will terminate the record and punch a trailer tape. When reading a record, the reader will stop at the EOT character. If no EOT character is present, the reader must be manually turned off and the Reset switch must be pressed to enter the operating system program.

C

Cold start at address 0100H in the system. Presently this has the effect of executing the extended interrupt handler that was loaded into RAM from ROM during power-up. Data from the optical shaft encoders and the four sonar range boards are sent thru the terminal serial port of the navigation computer to whatever device is connected to it at 9600 baud. (The extended interrupt handler, when active, will respond to the following commands:

Control S Stop sending data.
Control Q Resume sending data.
Control C Restart Nav computer.
1 Select sonars 1 at a time. 2 Select sonars 2 at a time. 3 Select sonars 3 at a time. 4 Select sonars 4 at a time.) If the ROM located at U7 is replaced with a ROM containing a 6800 BASIC interpreter then a cold start of BASIC is made.

W

Warm start at address 0103H in the system. This command is only effective if the ROM located at U7 is replaced with a ROM containing a 6800 BASIC interpreter. However, PROTO will jump to address 0103H any time this command is given irreguardless of the contents of that location.

O

Odyssey command. Causes PROTO to jump to address 02001H in the system.

INTERRUPTS

Of the four available interrupt vectors, RESET and SWI are used by PROTO. IRQ is used by the Sonar/Optical Shaft Encoder/Tic Time system. NMI is not used and is available for future designs. All interrupt vectors are in RAM except RESET so that a program can reconfigure the interrupt structure. A hardware RESET always returns the user to the PROTO operating system after a hardware initialization and ROM to RAM overlay has been performed.

The upper memory locations are ROM. If the user expects either NMI or IRQ interrupts to occur, he may reinitialize the vector addresses to the starting address of his IRQ and NMI handler routines by changing addresses BFF7-BFFF.

PROTO must have control of the RESET vector so that the RESET switch on the Prototyping Board can return program control to PROTO at any time.

The reset routine copies the contents of the B, A, X, CC, and S registers into a fixed area of memory. This means that the program can be aborted at any time by using the reset switch while still saving all the registers except the program counter. Unfortunately, the contents of the program counter are lost.

It is possible for the user to use the NMI interrupt to abort a program execution without losing the contents of the P and C registers. This condition is automatically set in the NMI handling routine when PROTO is called. This interrupt vector will cause the contents of the user's registers to be printed when the $\overline{NMI}$ lines goes low.

Since the SWI instruction is used to call subroutines between 00 and H'18 from (RS)[3] as described later in "The Subroutine ROM", the user is somewhat limited in the ways he can use SWI instructions. However, he can access an SWI handler routine in his own program by an SWI instruction followed by a byte containing the decimal number less than H'80 but greater than H'19 < n < H'80 sequence, PROTO passes control at address BFF3. If the user expects to access his own SWI routine and use PROTO, he must use the Set Memory command to store the address of this routine at locations BFF3 and BFF4.

PROTO makes sure that the user's SWI routine is entered from the stack with all registers containing the same information that they would hold if the routine were entered directly through the SWI vector.

EXTENDED INTERRUPT HANDLER

The extended interrupt handler that was loaded into RAM from ROM during power-up sends data from the optical shaft encoders and the four sonar range boards thru the terminal serial port of the navigation computer to whatever device is connected to it at 9600 baud. The extended interrupt handler becomes active when the Cold command is given from PROTO.The Nav computer will then respond to the following commands ONLY:

Control S   Stop sending data.

Control Q   Resume sending data.

Control C   Restart Nav computer.

1   Select sonars 1 at a time.   2   Select sonars 2 at a time.   3   Select sonars 3 at a time.   4   Select sonars 4 at a time.

THE SUBROUTINE ROM

Many of the monitor's functions are accomplished with the help of the Re-Entrant Self-Relative Subroutine ROMs $(RS)^3$. This standard ROM, which can be considered a software extension to the 6800 instruction set, is also available to be used by the user both on the prototype board and in his final production system. The user can call one of the 25 $(RS)^3$ subroutines with an SWI instruction followed by the number of the desired subroutine. The details of the subroutines available in the $(RS)^3$ user's manual.

The user should be aware of the fact that the $(RS)^3$ pushes from 7 to 10 bytes of data onto the stack, depending upon which subroutines are called. This means that if the user calls $(RS)^3$ routines, he must make sure that the necessary memory space is available for stack expansion.

Since PROTO assigns its own stack area, the user need not be concerned about how $(RS)^3$ is used.

BREAKPOINTS

Breakpoints allow the user to halt his program and examine the contents of the internal registers. PROTO provides two types of breakpoints. In this system, breakpoints are actually debugging routines that can be called from the user's program just like $(RS)^3$ routines.

Each breakpoint requires a two byte calling sequence: an SWI instruction followed by a number.

Breakpoints may be inserted either by reassembling the program with the extra SWI instructions added the Set Memory command may be used to replace parts of the code with SWI instructions. Note that the second method is not satisfactory for the snapshot option (described below) since the replaced code must be restored before execution can be continued. When using the second method, the user must make sure that he replaces the first two bytes of an instruction. If the SWI replaces the second or third byte of an instruction, it may be interpreted as an address rather than an opcode.

The different types of breakpoints are:

1. Print registers (SWI, H'80)
2. Snapshot (SWI, H'81)

The sequence SWI, H'80 saves the user's registers at the vector stored in BFF1-BFF2, prints their contents (in the order CC BB AA XXXX PPPP SSSS), then returns control to PROTO.

The sequence SWI, H'81 prints out the contents of the user's registers then continues executing the user's program starting at the address following the byte containing the number H'81. Note that if this address does not contain a valid opcode, unpredictable results will occur.

Reentrant Self-Relative Subroutine ROMs (RSRSRS) = $(RS)^3$

The cost of microprocessor software development is many small items: the cost of assembly time, storage time, transmission time, loading time, design, development, documentation and debug. The cost of many of these items continues to accumulate even though a subroutine library exists for common functions, in particular the time and cost of transmission, loading and ROM pattern generation.

The purpose of Reentrant Self-Relative Subroutine ROMs $(RS)^3$ is to give the user a hardware subroutine package which exists in the breadboard design from the beginning. The programs are documented, debugged and constitute some of the most commonly performed subroutines that assembly language programmers generate. The subroutines are not complex and are not intended to be. Any subroutine could easily be reproduced by a user; however, the intention is that the routine exists now and the user does not have to reproduce it. The routines tend to be slow because of their generality but the intention is immediate availability. If a particular program is time critical, it can be regenerated later when the time critical elements are known

CONCEPTS

The $(RS)^3$ uses a number of concepts to allow flexibility in the user environment. The first concept is self-relative programming. This simply means that the program will function correctly regardless of where it is located in memory. The user will need to know where it is located so he can reference it. However, this actual location will only have to be recorded once. The self-relative program uses relative address instructions for program control and the index and stack pointer instructions for data manipulation.

The stack is used for temporary storage of data to prevent $(RS)^3$ from being tied to fixed addresses. This allows the program to be reentrant; i.e. the program can be called at different times without completing the previous call. This means that the same routine can be called by the interrupt processor as well as by the program which was interrupted. The concept of reentrant code is not to be confused with recursive code; even through recursive coding could have been used in the subroutine package, it is not.

The subroutine calling mechanism uses the SWI instruction followed by a single byte index for the particular subroutine invoked. This was chosen because the SWI is the most convenient from an internal programming viewpoint and the safest. It is safe because an error in a ROM can be corrected by replacing the subroutine ROM without altering any other user ROM. If direct addresses to subroutine code exist in the user's domain, his ROMs would change if the location of the routine in the $(RS)^3$ changed.

IMPLEMENTATION

The user places the base address of the $(RS)^3$ into the SWI vector address. Each SWI instruction requires an index byte to follow the SWI instruction where the index indicates the function to be executed. After the function is performed, the user program will continue with the instruction following the index byte. In essence, a whole new set of instructions have been created for the user which are two bytes long.

To make the entry easier, a macro call can be provided which will assemble the correct index byte when the function name is used. A set of EQU assembler commands associates the name and the index byte value.

(RS)³ IN PROTO – SUBROUTINE DESCRIPTIONS

Each of the subroutines in the ROM are described here, giving the index for the call, a mnemonic subroutine name, a descriptive title, and the number of bytes in the stack used by the call (including the SWI). A brief description of the subroutine operation is also given, with the entry requirements, the exit conditions, and the registers altered by the subroutine. Only those registers indicated are altered by any (RS)³ subroutine.

Index	Name	Title	Stack Bytes
00	PUSH ALL	Push All Registers	14

Five bytes are pushed onto the stack, containing, respectively, the Condition Codes, the B and A accumulators, and the Index Register. No registers are altered (except the stack pointer, which is decremented by 5).

Entry: Any

Exit: Stack: SP +1 +2 +3 +4 +5 (=old SP)
CC, B, A, XHXL

Registers Altered: SP

Index	Name	Title	Stack Bytes
01	POPALL	Pop (=Pull) All Registers	9

Five bytes are pulled from the stack into the Condition Codes, the B and A accumulators, and the Index register, respectively. The Stack Pointer is incremented by 5.

Entry: Stack, as by PUSH ALL

Exit: CC, B, A, X pulled from stack

Registers Altered: CC, B, A, X, SP

Index	Name	Title	Stack Bytes
02	TXAB	Transfer Index Register to A and B	9

The most significant eight bits of the index register are copied to the A accumulator, and the least significant eight bits are copied to the B accumulator.

Entry: Any

Exit: A, B loaded from X

Registers Altered: A, B

Index	Name	Title	Stack Bytes
03	TABX	Transfer A and B to Index	9

Accumulator A is copied to the most significant byte position of the index register, and accumulator B is copied to the least significant byte position of the index register.

Entry: Any

Exit: X loaded from A, B

Registers Altered: X

Index	Name	Title	Stack Bytes
04	XABX	Exchange A and B with Index	12

The contents of the Index register and the two accumulators are exchanged, A with the most significant byte of X, B with the least significant byte.

Entry: Any

Exit: A, B and X exchanged

Registers Altered: A, B, X

Index	Name	Title	Stack Bytes
05	PUSHX	Push Index Register	11

The contents of the index register is pushed onto the stack. The Stack Pointer is decremented by two.

Entry: Any

Exit: Stack: SP +1 +2 (=old SP)
XH XL

Registers Altered: SP

Index	Name	Title	Stack Bytes
06	PULLX	Pop (=Pull) Index Register from Stack	9

Two bytes are pulled from the stack into the index register, and the stack pointer is incremented by two.

Entry: Two bytes on stack

Exit: X pulled from stack

Registers Altered: X, SP

Index	Name	Title	Stack Bytes
07	ADDXAB	Add Index to A and B	14

Add the contents of the Index Register to the two accumulators, as a 16-bit sum, leaving the result in the two accumulators. The most significant byte is assumed to be in accumulator A. The condition codes are set according to the result.

Entry: Addend in X, augend in A, B

Exit: Sum in A, B

Condition Codes:
- H = carry from bit 11 to bit 12 of sum
- N = bit 15 of sum
- Z = 1 if sum is zero; else = 0
- V = 1 if two's complement overflow
- C = carry out of bit 15 of sum

Registers Altered: A, B, CC

Index	Name	Title	Stack Bytes
08	ADDABX	Add A and B to Index Register	9

Add the contents of the two accumulators to the Index register, leaving the 16-bit sum in the index register. Accumulator A is assumed to be more significant than accumulator B. The condition codes are set according to the result.

Entry: Addend in A, B; augend in X

Exit: Sum in X

Condition Codes:
- H = carry from bit 11 to bit 12 of sum
- N = bit 15 of sum
- Z = 1 if sum is zero, =0 otherwise
- V = 1 if two's complement overflow
- C = carry out of bit 15 of sum

Registers Altered X, CC

Index	Name	Title	Stack Bytes
09	ADDAX	Add A to Index Register	9

Add the A accumulator to the contents of the Index register, and return the sum to the index register. The Condition Codes are set according to the result.

Entry: Addend in A, augend in X

Exit: Sum in X

Condition Codes: (Same as ADDABX)

Registers Altered: X, CC

Index	Name	Title	Stack Bytes
0A	ADDBX	Add B to Index Register	9

Add the contents of the B accumulator to the Index register, and leave the sum in the Index register. The Condition Codes are set according to the result.

Entry: Addend in B, augend in X

Exit: Sum in X

Condition Codes: (Same as ADDABX)

Registers Altered: X, CC

Index	Name	Title	Stack Bytes
0B	SUBXAB	Subtract Index from A, B	14

Subtract the contents of the index register from accumulators A and B as a 16-bit difference. The Condition Codes are set according to the result.

Entry: Subtrahend in X, minuend in A, B

Exit: Difference in A, B

Condition Codes:
- H = undefined
- N = bit 15 of difference
- Z = 1 if result is zero, =0 otherwise
- V = 1 if two's complement overflow
- C = borrow into bit 15 of difference

Registers Altered: A, B, CC

Index	Name	Title	Stack Bytes
0C	SUBABX	Subtract A and B from Index Register	9

Subtract the contents of the A and B accumulators from the Index register, leaving the difference in the Index. The Condition Codes are set according to the result.

Entry: Subtrahend in A, B; minuend in X

Exit: Difference in X

Condition Codes: (Same as SUBXAB)

Registers Altered: X, CC

Index	Name	Title	Stack Bytes
0D	SUBAX	Subtract A from Index Register	9

Subtract the contents of the A accumulator from the contents of the Index register and return the difference to the index register. The Condition Codes are set according to the result.

Entry: Subtrahend in A, minuend in X

Exit: Difference in X

Condition Codes: (Same as SUBXAB)

Registers Altered: X, CC

Index	Name	Title	Stack Bytes
0E	SUBBX	Subtract B from Index Register	9

Subtract the contents of the B accumulator from the Index register, leaving the difference in the index register. The Condition Codes are set according to the result.

Entry: Subtrahend in B, minuend in X

Exit: Difference in X

Condition Codes: (Same as SUBXAB)

Registers Altered: X, CC

Index	Name	Title	Stack Bytes
0F	P2HEX	Print Byte in Hex	15

The byte pointed to by the address in the Index register is converted to hexadecimal notation in ASCII, and output to the ACIA located as follows: memory locations BFF5–BFF6 contain an address of a pair of bytes (indirect pointer) which in turn contain the address of the ACIA status register.

BFF6	iL
BFF5	iH
...	
i+1	aL
i	aH
...	
a+1	ACIA Data
a	ACIA Status

Each byte of the output is stored into the ACIA data register after bit 1 of the Status register is true. The Control register of the ACIA is not altered, and the Data register is not read by this routine. The Index register is incremented past the byte which is output.

Entry: Memory byte at (X); ACIA at (BFF5)

Exit: (two ASCII bytes output)

Registers Altered: X

Index	Name	Title	Stack Bytes
10	P4HEX	Print Address in Hex	15

The two bytes in memory pointed to by the Index register are converted to four ASCII digits and output to the ACIA located at the address pointed to by the pointer pointed to by the byte pair at BFF5–BFF6 (see P2HEX). The Index register is incremented by two.

Entry: Two bytes at (X); ACIA at ((BFF5))

Exit: (four ASCII bytes output)

Registers Altered: X

Index	Name	Title	Stack Bytes
11	PRINTA	Print the Byte in A	10

The byte in accumulator A is output to the ACIA, the address of whose address is in locations BFF5–BFF6. No registers are altered except the ACIA data register.

Entry: Character in A

Exit: (one byte output)

Registers Altered: None

Index	Name	Title	Stack Bytes
12	PMSG	Print Message String	12

A message string, the first byte of which is pointed to by the Index register, is output to the ACIA, the address of whose address is in locations BFF5 BFF6. The string is terminated by an ASCII EXT (=hex 04), and the Index register is left pointing to that byte on return.

Entry: Character string to (X) terminated by 04; ACIA at ((BFF5))

Exit: (in ASCII bytes output), X pointing to 04 byte

Registers Altered: X

Index	Name	Title	Stack Bytes
13	VALAN	Validate AlphaNumeric	11

The character pointed to by the Index register is analyzed, and the Carry flag is set if it is a letter or digit; if it is not a hexadecimal digit, the Overflow flag is set. Other than the condition codes, no registers are altered.

Entry: Memory byte (ASCII) at (X)

Exit: Condition Codes:

H = undefined
N = undefined
Z = 0
V = 0 if character in range 0–9, A–F; else = 1
C = 1 if character in range 0–9, A–Z; else = 0

Registers Altered: CC

Index	Name	Title	Stack Bytes
14	INPUTA	Input ACIA byte to A	9

One byte is input from the ACIA, the address of whose address is at location BFF5-BFF6 , and this byte is returned to accumulator A. The ACIA is not written to, and except for the A accumulator, no registers are changed. (RS)[3] samples bit 0 of the status register of the ACIA, and when it goes to one, reads the data register. The input byte has bit 7 removed (set to zero).

Entry: (one byte input)

Exit: Character in A, bit 7=0

Registers Altered: A

Index	Name	Title	Stack Bytes
15	CONHB	Convert Hex String to Binary	11

A string of characters in memory beginning at the address in the index register is scanned for valid Hexadecimal digits; when one is found, it and all immediately following hex digits are converted to a binary number, which is left in the A and B accumulators (A is more significant). When this routine is called, the maximum length of the string is in the B accumulator. On exit, the Carry flag is set to one if the conversion resulted in a valid binary number, and the index register is left pointing to the next character in the string, or if the string is exhausted before finding any hex digits, to the last character of the string.

Entry: Character string (including ASCII hex number) at (X)
Max string length in B (<128)

Exit: Binary number in A, B

Condition Codes:

H = undefined
N = undefined
Z = undefined
V = undefined
C = 1 if valid number; = 0 if not

Registers Altered: A, B, X, CC

Index	Name	Title	Stack Bytes
16	INDEX	Multiply A X B and Add to Index	12

The contents of the A accumulator is multiplied by the contents of the B accumulator, and the product is added to the Index register. The Condition Codes are set according to the result.

Entry: Multiplicand in A, Multiplier in B, augend in X

Exit: Sum in X

Condition Codes: (Same as ADDABX)

Index	Name	Title	Stack Bytes
17	MUL8	Multiply A Times B	12

Multiply the contents of the A accumulator times the contents of the B accumulator, and leave the product in both accumulators as a 16-bit number, with the most significant part in A. This is an unsigned multiply, and if either or both of the factors is negative (two's complement signed) the product will not be a true signed product of the signed factors, as may be seen in this formula:

$$(-n)\times(m) = (256 - n)\times m = 256m + (-nm)$$

The condition codes are nonetheless set according to the result.

Entry: Multiplicand in A, multiplier in B

Exit: Product in A, B

Condition Codes:
- H = undefined
- N = bit 15 of product
- V = 0
- Z = 1 if product is zero; otherwise = D
- C = 0

Registers Altered: A, B, CC

<u>Appendix O</u>

Data from the Optical Shaft Encoder and Ultrasonic Sonar subsystems are gathered every 0.1 second via the Extended Interrupt Handler (see Appendix B) and stored into a temporary line buffer in the Navigation Computer's memory as follows:

/time/fw/lw1/lw2/rw1/rw2/A#____/B#____/C#____/D#____/<CR><LF>

Where time = tenth's of seconds count - 2 bytes

fw = front wheel direction - 1 byte

lw1 = left wheel reverse counts - 2 bytes

lw2 = left wheel forward counts - 2 bytes

rw1 = right wheel reverse counts - 2 bytes

rw2 = right wheel forward counts - 2 bytes

A, B, C, or D refer to Sonar transducer selected and range of closest object except where an astrisk (*) indicates no transducer selected.

All numbers are hexadecimal except the sonar ranges which are decimal feet.

Test Data for Robot Integrated Operation Test # 20

```
/0001/49/0000/0000/0000/0001/A2 7.2/B*15.1/C0 2.5/D1 9.4/
/0002/49/0000/0000/0000/0001/A019.3/B1 4.7/C2 3.9/D*10.7/
/0003/49/0000/0000/0000/0001/A111.2/B2 2.9/C*11.3/D0 7.1/
/0004/49/0000/0000/0000/0001/A2 7.4/B*13.7/C0 2.4/D1 8.4/
/0005/49/0000/0000/0000/0001/A*12.6/B0 6.1/C1 3.0/D2 9.9/
```

```
/0006/49/0000/0000/0000/0001/A411.6/B5 3.5/C6 6.9/D*10.2/
/0007/49/0000/0000/0000/0001/A5 8.2/B6 2.6/C*11.9/D4 7.3/
/0008/49/0000/0000/0000/0001/A6 6.2/B*14.7/C4 2.7/D5 9.0/
/0009/49/0000/0000/0000/0001/A*17.8/B4 6.4/C5 3.5/D610.0/
/000A/49/0000/0000/0000/0001/A014.2/B1 4.9/C2 3.8/D*10.1/
/000B/49/0000/0000/0000/0001/A111.2/B2 2.9/C*11.3/D0 7.0/
/000C/47/0000/0000/0000/0001/A2 7.2/B*14.9/C0 2.5/D1 7.4/
/000D/47/0000/0000/0000/0001/A*12.9/B0 6.2/C1 3.0/D210.2/
/000E/48/0000/0000/0000/0001/A411.7/B5 3.4/C6 7.0/D* 9.9/
/000F/48/0000/0000/0000/0001/A5 8.1/B6 2.7/C*14.3/D4 7.3/
/0010/48/0000/0000/0000/0001/A6 6.3/B*13.1/C4 2.8/D5 9.1/
/0011/48/0000/0000/0000/0001/A*17.5/B4 6.4/C5 3.4/D611.2/
/0012/48/0000/0000/0000/0002/A013.1/B1 4.8/C2 3.8/D*10.1/
/0013/48/0000/0000/0000/0002/A111.2/B2 3.0/C*11.3/D0 7.1/
/0014/48/0000/0001/0000/0002/A2 7.3/B*14.8/C0 2.5/D1 7.3/
/0015/48/0000/0001/0000/0003/A*14.8/B0 6.1/C1 3.1/D2 8.5/
/0016/48/0000/0002/0000/0003/A411.6/B5 3.6/C6 7.3/D* 8.2/
/0017/48/0000/0003/0000/0004/A5 8.2/B6 2.9/C*14.7/D4 7.2/
/0018/47/0000/0004/0000/0005/A6 6.4/B*12.2/C4 2.9/D5 9.1/
/0019/47/0000/0005/0000/0006/A*17.4/B4 6.4/C5 3.8/D610.4/
/001A/47/0000/0006/0000/0008/A013.4/B1 5.7/C2 5.3/D*10.2/
/001B/47/0000/0007/0000/0009/A110.7/B2 3.4/C*11.4/D0 7.0/
/001C/47/0000/0009/0000/000A/A2 7.3/B*13.0/C0 3.2/D1 7.4/
/001D/47/0000/000A/0001/000B/A*15.5/B0 6.2/C1 3.6/D2 8.5/
/001E/47/0000/000B/0001/000D/A414.5/B5 4.6/C6 6.8/D*10.0/
/001F/47/0000/000D/0001/000E/A5 8.8/B6 3.7/C*14.1/D4 7.3/
/0020/47/0000/000E/0001/000F/A6 6.3/B*11.5/C4 3.6/D510.2/
/0021/46/0000/000F/0001/0011/A*14.7/B4 6.5/C5 4.8/D611.2/
/0022/46/0000/0011/0001/0012/A012.8/B1 6.8/C2 6.6/D*10.1/
/0023/46/0000/0012/0001/0013/A110.1/B2 4.4/C*10.1/D0 7.1/
/0024/46/0000/0013/0001/0015/A2 7.1/B*12.9/C0 4.0/D1 7.3/
/0025/46/0000/0015/0001/0016/A*12.8/B0 6.2/C1 4.7/D2 8.1/
/0026/46/0000/0016/0001/0017/A414.7/B5 5.7/C6 7.1/D* 9.9/
/0027/46/0000/0017/0001/0019/A5 9.6/B6 4.6/C*14.5/D4 7.3/
/0028/46/0000/0018/0001/001A/A6 6.4/B*14.6/C4 4.7/D510.1/
/0029/46/0000/001A/0001/001B/A*17.6/B4 6.3/C5 6.3/D610.7/
/002A/45/0000/001B/0002/001C/A011.1/B1 6.9/C2 7.1/D* 9.0/
/002B/45/0000/001C/0002/001E/A112.9/B2 5.4/C*11.6/D0 7.1/
/002C/45/0000/001E/0002/001F/A2 6.8/B*14.0/C0 5.0/D1 7.5/
/002D/45/0000/001F/0002/0020/A*17.1/B0 6.2/C1 5.5/D210.0/
/002E/45/0000/0020/0002/0022/A417.5/B5 6.8/C6 7.0/D* 9.9/
/002F/44/0000/0022/0002/0023/A5 8.9/B6 5.5/C*14.4/D4 7.4/
/0030/44/0000/0023/0002/0024/A6 6.4/B*14.5/C4 5.5/D5 9.5/
/0031/44/0000/0024/0002/0025/A*15.1/B4 6.5/C5 7.2/D611.4/
/0032/44/0000/0025/0002/0027/A014.9/B1 7.1/C2 6.6/D*10.0/
/0033/44/0000/0027/0002/0028/A111.8/B2 6.2/C*11.8/D0 7.1/
/0034/44/0000/0028/0003/0029/A2 6.8/B*13.0/C0 5.8/D1 7.5/
/0035/44/0000/0029/0003/002B/A*12.8/B0 6.2/C1 6.3/D210.2/
/0036/43/0000/002A/0003/002C/A416.6/B5 7.9/C6 7.0/D* 9.8/
/0037/43/0000/002C/0003/002D/A5 8.3/B6 6.3/C*14.0/D4 7.3/
/0038/43/0000/002D/0003/002E/A6 6.4/B*12.3/C4 6.6/D5 9.1/
/0039/43/0000/002E/0003/0030/A*18.0/B4 6.6/C5 6.6/D6 9.8/
```

```
/003A/43/0000/0030/0003/0031/A010.3/B1 6.8/C2 6.9/D*10.3/
/003B/43/0000/0031/0003/0032/A111.5/B2 7.1/C*11.7/D0 7.1/
/003C/43/0000/0032/0003/0033/A216.9/B*13.0/C0 6.5/D1 8.8/
/003D/43/0000/0033/0003/0035/A*17.4/B0 6.3/C1 6.7/D2 8.5/
/003E/42/0000/0035/0004/0036/A4 9.6/B5 8.5/C6 7.3/D* 9.7/
/003F/42/0000/0036/0004/0037/A5 7.9/B6 7.2/C*14.3/D4 7.3/
/0040/42/0000/0037/0004/0038/A6 6.4/B*11.8/C4 7.0/D5 9.0/
/0041/42/0000/0038/0004/003A/A*17.9/B4 6.5/C5 7.9/D610.3/
/0042/42/0000/003A/0004/003B/A014.6/B1 7.2/C2 6.6/D*10.2/
/0043/42/0000/003B/0004/003C/A110.8/B2 7.9/C*13.3/D0 7.1/
/0044/42/0000/003C/0004/003E/A2 7.2/B*14.6/C0 6.6/D1 8.2/
/0045/41/0000/003D/0004/003F/A*18.0/B0 6.1/C1 6.7/D210.2/
/0046/41/0000/003F/0004/0040/A410.4/B5 9.6/C6 7.3/D*10.1/
/0047/41/0000/0040/0004/0041/A510.4/B6 8.0/C*14.8/D4 7.4/
/0048/41/0000/0041/0004/0043/A6 6.3/B*14.6/C4 8.1/D510.4/
/0049/41/0000/0042/0005/0044/A*17.6/B4 6.5/C5 6.9/D612.5/
/004A/41/0000/0044/0005/0045/A014.2/B1 7.3/C2 7.5/D*10.3/
/004B/41/0000/0045/0005/0046/A111.3/B2 8.7/C*11.3/D0 7.1/
/004C/41/0000/0046/0005/0048/A2 7.0/B*14.4/C0 6.7/D1 7.8/
/004D/40/0000/0047/0005/0049/A*17.4/B0 6.2/C1 6.7/D2 8.4/
/004E/40/0000/0049/0005/004A/A414.0/B5 8.1/C6 7.3/D*10.1/
/004F/40/0000/004A/0005/004B/A5 8.7/B6 8.7/C*14.5/D4 7.5/
/0050/40/0000/004B/0005/004D/A6 6.3/B* 9.9/C4 8.1/D5 9.8/
/0051/40/0000/004C/0005/004E/A*16.7/B4 6.3/C5 8.2/D611.5/
/0052/40/0000/004E/0005/004F/A0 9.8/B1 7.3/C2 7.3/D* 9.9/
/0053/40/0000/004F/0006/0051/A111.4/B2 8.8/C*12.6/D0 7.1/
/0054/40/0000/0050/0006/0052/A2 6.7/B*14.4/C0 6.8/D1 8.0/
/0055/40/0000/0052/0006/0053/A*17.4/B0 6.2/C1 6.6/D210.2/
/0056/40/0000/0053/0006/0054/A413.3/B5 8.5/C6 7.2/D*10.6/
/0057/3F/0000/0054/0006/0056/A5 9.0/B6 9.7/C*14.9/D4 7.4/
/0058/3F/0000/0055/0006/0057/A6 6.4/B*14.2/C4 8.1/D5 9.3/
/0059/3F/0000/0057/0006/0058/A*17.3/B4 6.5/C5 6.8/D611.9/
/005A/3F/0000/0058/0007/0059/A012.5/B1 9.9/C2 7.1/D*10.0/
/005B/3E/0000/0059/0007/005B/A111.4/B2 8.6/C*11.4/D0 7.3/
/005C/3E/0000/005A/0007/005C/A2 7.2/B*14.6/C0 6.7/D1 9.0/
/005D/3E/0000/005C/0007/005D/A*17.9/B0 6.2/C1 6.7/D2 9.7/
/005E/3D/0000/005D/0007/005E/A412.3/B5 9.1/C6 7.3/D* 9.8/
/005F/3D/0000/005E/0007/0060/A5 8.5/B6 8.4/C*13.5/D4 7.3/
/0060/3D/0000/005F/0007/0061/A6 6.1/B*14.6/C4 8.2/D5 9.5/
/0061/3D/0000/0061/0007/0062/A*17.9/B4 6.5/C5 6.6/D611.7/
/0062/3C/0000/0062/0007/0063/A011.6/B1 6.8/C2 6.8/D*10.4/
/0063/3C/0000/0063/0008/0065/A111.3/B2 9.7/C*13.3/D0 7.1/
/0064/3C/0000/0064/0008/0066/A2 7.2/B*14.7/C0 6.7/D1 8.6/
/0065/3C/0000/0066/0008/0067/A*12.8/B0 6.2/C1 6.9/D210.2/
/0066/3C/0000/0067/0008/0069/A411.7/B5 9.6/C6 7.5/D* 9.8/
/0067/3C/0000/0068/0008/006A/A5 7.9/B611.3/C*14.4/D4 7.4/
/0068/3C/0000/006A/0008/006B/A6 6.3/B*12.3/C4 8.2/D5 8.1/
/0069/3C/0000/006B/0008/006C/A*18.4/B4 6.5/C5 8.1/D611.2/
/006A/3C/0000/006C/0008/006E/A010.7/B1 7.1/C2 7.2/D*10.7/
/006B/3C/0000/006D/0008/006F/A110.6/B2 9.8/C*13.1/D0 7.1/
/006C/3C/0000/006F/0009/0070/A2 7.2/B*14.4/C0 6.7/D1 8.2/
/006D/3B/0000/0070/0009/0071/A*14.5/B0 5.8/C1 6.8/D210.9/
```

```
/006E/3B/0000/0071/0009/0073/A410.7/B510.7/C6 7.7/D*10.7/
/006F/3B/0000/0072/0009/0074/A512.4/B612.1/C*14.6/D4 7.4/
/0070/3B/0000/0074/0009/0075/A6 6.4/B*11.0/C4 8.1/D510.9/
/0071/3B/0000/0075/0009/0076/A*18.2/B4 6.6/C5 8.2/D610.3/
/0072/3B/0000/0076/0009/0078/A010.0/B1 7.5/C2 8.1/D*10.9/
/0073/3B/0000/0077/0009/0079/A110.0/B2 9.7/C*14.0/D0 7.2/
/0074/3B/0000/0079/0009/007A/A2 7.1/B*13.2/C0 6.7/D1 7.9/
/0075/3B/0000/007A/0009/007B/A*17.5/B0 6.1/C1 7.2/D210.2/
/0076/3A/0000/007B/0009/007D/A410.1/B5 8.0/C6 7.4/D*10.1/
/0077/3A/0000/007C/000A/007E/A5 9.7/B6 9.7/C*14.2/D4 7.5/
/0078/3A/0000/007E/000A/007F/A6 6.3/B*14.5/C4 8.3/D510.5/
/0079/3A/0000/007F/000A/0080/A*18.0/B4 6.2/C5 8.1/D6 9.6/
/007A/3A/0000/0080/000A/0082/A0 9.1/B1 7.3/C2 7.8/D* 9.3/
/007B/3A/0000/0081/000A/0083/A1 9.8/B2 9.7/C* 9.9/D0 7.3/
/007C/3A/0000/0082/000A/0084/A2 6.7/B*15.2/C0 6.7/D1 8.1/
/007D/3A/0000/0084/000A/0085/A*17.1/B0 6.1/C1 7.2/D210.3/
/007E/3A/0000/0085/000A/0086/A4 9.3/B5 8.4/C6 7.3/D*11.3/
/007F/3A/0000/0086/000A/0088/A5 8.7/B611.2/C*14.9/D4 7.5/
/0080/3A/0000/0087/000A/0089/A6 6.4/B*12.2/C4 8.2/D5 9.9/
/0081/3A/0000/0089/000B/008A/A*18.8/B4 6.2/C5 8.2/D6 8.7/
/0082/3A/0000/008A/000B/008B/A0 8.3/B1 9.1/C2 7.9/D* 9.8/
/0083/39/0000/008B/000B/008D/A110.1/B210.0/C*10.1/D0 7.4/
/0084/39/0000/008C/000B/008E/A2 6.9/B*14.7/C0 6.7/D1 9.1/
/0085/39/0000/008E/000B/008F/A*17.3/B0 6.1/C1 6.8/D2 8.5/
/0086/39/0000/008F/000B/0090/A4 8.6/B5 9.0/C6 7.3/D*10.5/
/0087/39/0000/0090/000B/0092/A5 8.4/B6 8.4/C*14.4/D4 7.6/
/0088/39/0000/0091/000B/0093/A6 6.2/B*15.4/C4 8.2/D5 9.5/
/0089/39/0000/0093/000B/0094/A*15.7/B4 6.5/C5 8.3/D6 8.0/
/008A/38/0000/0094/000C/0095/A0 7.5/B1 6.6/C2 7.7/D*11.3/
/008B/38/0000/0095/000C/0096/A1 8.5/B2 9.8/C*14.6/D0 7.4/
/008C/38/0000/0096/000C/0098/A2 7.2/B*15.0/C0 7.2/D1 9.2/
/008D/37/0000/0097/000C/0099/A*20.6/B0 6.1/C1 7.9/D2 8.4/
/008E/37/0000/0099/000C/009A/A4 7.8/B5 9.4/C6 7.5/D*11.3/
/008F/37/0000/009A/000C/009B/A5 7.8/B610.1/C*15.3/D4 7.7/
/0090/37/0000/009B/000C/009C/A6 6.1/B*16.0/C4 8.1/D510.2/
/0091/37/0000/009C/000C/009E/A*20.5/B4 6.4/C5 8.1/D6 7.0/
/0092/37/0000/009D/000C/009F/A0 6.7/B1 6.9/C2 7.0/D*11.3/
/0093/37/0000/009E/000D/00A0/A1 7.5/B210.3/C*14.4/D0 7.4/
/0094/37/0000/00A0/000D/00A1/A2 7.1/B*14.6/C0 7.1/D1 8.7/
/0095/37/0000/00A1/000D/00A2/A*17.9/B0 6.1/C1 7.4/D210.0/
/0096/37/0000/00A2/000D/00A3/A4 6.7/B5 9.0/C6 7.5/D*10.9/
/0097/37/0000/00A3/000D/00A5/A5 7.6/B614.5/C*14.5/D4 7.7/
/0098/37/0000/00A4/000D/00A6/A6 6.5/B*14.7/C4 8.2/D5 9.7/
/0099/37/0000/00A5/000E/00A7/A*19.6/B4 6.4/C5 8.2/D6 6.4/
/009A/37/0000/00A7/000E/00A8/A0 5.9/B1 7.3/C2 8.3/D*10.8/
/009B/37/0000/00A8/000E/00A9/A1 6.7/B210.2/C*14.4/D0 7.4/
/009C/37/0000/00A9/000E/00AA/A2 7.1/B*14.4/C0 6.7/D1 8.4/
/009D/37/0000/00AA/000E/00AC/A*16.9/B0 6.0/C1 6.7/D2 6.2/
/009E/37/0000/00AB/000E/00AD/A410.6/B5 7.8/C6 7.5/D*10.7/
/009F/37/0000/00AD/000E/00AE/A5 6.7/B610.0/C*15.6/D4 7.7/
/00A0/37/0000/00AE/000E/00AF/A6 6.5/B*15.5/C4 8.2/D5 9.4/
/00A1/37/0000/00AF/000E/00B0/A*20.5/B4 6.3/C5 8.3/D6 5.5/
```

```
/00A2/36/0000/00B0/000F/00B2/A0 5.2/B1 7.2/C2 8.5/D*11.7/
/00A3/36/0000/00B1/000F/00B3/A1 6.0/B210.1/C*14.6/D0 7.4/
/00A4/36/0000/00B2/000F/00B4/A2 6.7/B*15.3/C0 7.3/D1 8.2/
/00A5/36/0000/00B4/000F/00B5/A*17.2/B0 6.1/C1 7.8/D2 5.4/
/00A6/36/0000/00B5/000F/00B6/A4 5.0/B5 8.1/C6 7.6/D*11.3/
/00A7/36/0000/00B6/000F/00B8/A5 7.7/B614.5/C*14.6/D4 7.7/
/00A8/36/0000/00B7/000F/00B9/A6 6.3/B*14.5/C4 8.1/D5 9.1/
/00A9/36/0000/00B9/000F/00BA/A*20.7/B4 6.2/C5 8.5/D6 4.9/
/00AA/36/0000/00BA/000F/00BB/A0 4.3/B1 7.4/C2 8.2/D*10.2/
/00AB/36/0000/00BB/0010/00BD/A1 5.3/B2 9.0/C*14.8/D0 7.4/
/00AC/36/0000/00BC/0010/00BE/A2 6.4/B* 8.2/C0 8.1/D1 8.1/
/00AD/35/0000/00BE/0010/00BF/A*16.8/B0 6.0/C1 6.7/D2 4.7/
/00AE/35/0000/00BF/0010/00C0/A4 4.2/B5 8.7/C6 7.7/D*10.3/
/00AF/35/0000/00C0/0010/00C2/A5 7.4/B615.0/C*15.0/D4 7.6/
/00B0/35/0000/00C1/0010/00C3/A6 6.2/B*12.1/C4 8.5/D5 8.9/
/00B1/35/0000/00C3/0010/00C4/A*20.8/B4 6.3/C5 8.1/D6 7.6/
/00B2/34/0000/00C4/0010/00C6/A0 3.5/B1 6.6/C2 8.1/D*10.1/
/00B3/34/0000/00C5/0010/00C7/A1 5.0/B210.7/C*14.7/D0 7.5/
/00B4/34/0000/00C7/0010/00C8/A2 6.8/B*13.4/C0 7.6/D1 8.2/
/00B5/34/0000/00C8/0010/00C9/A*16.9/B0 6.1/C1 8.6/D2 8.2/
/00B6/33/0000/00C9/0010/00CB/A4 3.5/B5 9.4/C6 7.7/D*12.3/
/00B7/33/0000/00CA/0010/00CC/A5 4.2/B614.9/C*15.0/D4 7.8/
/00B8/33/0000/00CC/0011/00CD/A6 5.9/B*13.2/C410.0/D5 3.8/
/00B9/32/0000/00CD/0011/00CE/A*20.6/B4 6.3/C5 8.5/D6 5.4/
/00BA/32/0000/00CE/0011/00CF/A0 2.7/B1 6.9/C2 8.2/D* 9.0/
/00BB/32/0000/00CF/0011/00D1/A110.4/B210.0/C*13.5/D0 7.5/
/00BC/32/0000/00D0/0011/00D2/A2 6.5/B*15.1/C0 7.6/D1 7.9/
/00BD/31/0000/00D1/0011/00D3/A*17.6/B0 5.9/C1 8.2/D2 8.6/
/00BE/31/0000/00D2/0011/00D4/A412.1/B5 9.9/C6 7.9/D*10.1/
/00BF/31/0000/00D3/0011/00D4/A5 3.7/B615.0/C*15.0/D4 7.8/
/00C0/31/0000/00D3/0011/00D5/A6 6.4/B*12.5/C4 8.2/D5 3.3/
/00C1/31/0000/00D3/0011/00D5/A*18.0/B4 6.3/C5 8.4/D6 4.0/
/00C2/31/0000/00D4/0011/00D5/A0 2.1/B1 7.1/C2 8.2/D*11.7/
/00C3/31/0000/00D4/0011/00D5/A1 9.2/B2 9.1/C*14.5/D0 7.5/
/00C4/31/0000/00D4/0011/00D5/A2 6.4/B*13.2/C0 8.1/D1 8.1/
/00C5/31/0000/00D4/0011/00D5/A*20.0/B0 5.8/C1 8.1/D2 8.5/
/00C6/31/0000/00D4/0011/00D5/A4 2.3/B510.2/C6 7.8/D*11.5/
/00C7/31/0000/00D4/0011/00D5/A5 3.7/B614.4/C*14.4/D4 7.7/
/00C8/31/0000/00D4/0011/00D5/A6 6.2/B*12.6/C4 8.7/D5 3.4/
/00C9/31/0000/00D4/0011/00D5/A*18.3/B4 6.3/C5 9.3/D6 3.9/
/00CA/31/0000/00D4/0011/00D5/A0 2.2/B1 7.1/C2 8.5/D*11.5/
/00CB/31/0000/00D4/0011/00D5/A1 9.1/B2 9.0/C*14.5/D0 7.5/
/00CC/31/0000/00D4/0011/00D5/A2 6.5/B*13.0/C0 7.5/D1 8.1/
/00CD/31/0000/00D4/0011/00D5/A*17.5/B0 5.8/C1 8.2/D2 8.5/
/00CE/31/0000/00D4/0011/00D5/A4 2.4/B510.1/C6 7.8/D*10.1/
/00CF/31/0000/00D4/0011/00D5/A5 3.7/B617.8/C*14.5/D4 7.8/
/00D0/31/0000/00D4/0011/00D5/A6 6.3/B*14.1/C414.2/D5 3.3/
/00D1/31/0000/00D4/0011/00D5/A*20.6/B4 6.3/C5 9.7/D6 3.9/
/00D2/31/0000/00D4/0011/00D5/A0 2.1/B1 7.1/C2 8.4/D*14.8/
/00D3/31/0000/00D4/0011/00D5/A1 9.2/B2 9.1/C*14.8/D0 7.5/
/00D4/31/0000/00D4/0011/00D5/A2 6.4/B*14.8/C010.4/D1 8.2/
/00D5/31/0000/00D4/0011/00D5/A*19.9/B0 5.8/C111.3/D2 8.6/
```

```
/00D6/31/0000/00D4/0011/00D5/A4 2.3/B510.2/C6 7.8/D*14.8/
/00D7/31/0000/00D4/0011/00D5/A5 3.7/B618.0/C*18.1/D4 7.7/
/00D8/31/0000/00D4/0011/00D5/A6 6.2/B*14.1/C414.1/D5 3.4/
/00D9/31/0000/00D4/0011/00D5/A*20.5/B4 6.3/C5 9.6/D6 3.9/
/00DA/31/0000/00D4/0011/00D5/A0 2.2/B1 7.1/C2 8.5/D*14.9/
/00DB/31/0000/00D4/0011/00D5/A1 9.1/B2 9.1/C*14.7/D0 7.5/
/00DC/31/0000/00D4/0011/00D5/A2 6.5/B*14.8/C0 8.2/D1 8.1/
/00DD/31/0000/00D4/0011/00D5/A*19.8/B0 5.8/C111.3/D2 8.6/
/00DE/31/0000/00D4/0011/00D5/A4 2.3/B510.1/C6 7.9/D*14.7/
/00DF/31/0000/00D4/0011/00D5/A5 3.7/B615.1/C*15.0/D4 7.8/
/00E0/31/0000/00D4/0011/00D5/A6 6.2/B*14.1/C414.0/D5 3.3/
/00E1/31/0000/00D4/0011/00D5/A*20.6/B4 6.3/C5 9.7/D6 3.9/
/00E2/31/0000/00D4/0011/00D5/A0 2.1/B1 7.1/C2 8.3/D*13.0/
/00E3/31/0000/00D4/0011/00D5/A1 9.2/B2 9.0/C*14.8/D0 7.6/
/00E4/31/0000/00D4/0011/00D5/A2 6.4/B*17.8/C0 8.1/D1 8.1/
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT UNLIMITED
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/84D-19	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering — 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION — 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) Creating a Mobile Autonomous Robot Research System (MARRS)	

12. PERSONAL AUTHOR(S)
Thomas E. Clifford, 1st Lt, USAF   Hubert G. Schneider III, Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	1984 December	256

16. SUPPLEMENTARY NOTATION

Approved for public release: IAW AFR 190-17.
LYNN E. WOLAVER 28 Feb 85
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Autonomous Vehicle, Robot, Environmental Mapping, Basic, Sonar, Shaft Encoder, Ultrasonics, Navigation, Microprocessors, Computer Programs, Theses

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Mobile Autonomous Robot Research System (MARRS-1) was created as the first of a series of autonomous vehicle prototypes for the Air Force Institute of Technology. The major accomplishment in developing MARRS-1 is the integration of Optical Shaft Encoder (OSE) data with Ultrasonic Sonar range and direction information to produce accurate environmental maps that are relative to the robot. The OSE and Sonar subsystems make up the most important part of MARRS-1's Navigation Computer. With these two subsystems and minimal additional software, mapping and obstacle avoidance become a reality. The thesis includes schematics, parts list, and software listings for the MARRS-1 Navigation Computer. Additionally, the mapping and navigation algorithms are shown implemented in the BASIC language with numerous example graphics maps created by the integrated MARRS-1 robot. Issues involved in solving mobile robotics problems are discussed.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Matthew Kabrisky	22b. TELEPHONE NUMBER (Include Area Code) 513-255-5276 — 22c. OFFICE SYMBOL AFIT/ENG

DD FORM 1473, 83 APR   EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE